

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.

THIS PAGE BLANK (USPTO)



#4

#4

PATENT
80920.0015

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:

Pierre G. Richard

Serial No: 09/904,170

Filed: July 11, 2001

For: CONVERTING DATA HAVING
ANY OF A PLURALITY OF
MARKUP FORMATS AND A
TREE STRUCTURE

Art Unit: 2171

Examiner: Not Assigned

I hereby certify that this correspondence
is being deposited with the United States
Postal Service with sufficient postage as
first class mail in an envelope addressed
to: BOX MISSING PARTS

Commissioner for Patents
Washington, D.C. 20231, on

September 27, 2001

Date of Deposit

William H. Wright

Name

Signature

9/27/01

Date

TRANSMITTAL OF PRIORITY DOCUMENT

BOX MISSING PARTS

Commissioner for Patents

Washington, D.C. 20231

Dear Sir:

Enclosed herewith is a certified copy of French patent application
No. 0009105 which was filed July 12, 2000, from which priority is claimed under 35
U.S.C. § 119 and Rule 55.

Acknowledgment of the priority document(s) is respectfully requested to
ensure that the subject information appears on the printed patent.

Respectfully submitted,

HOGAN & HARTSON L.L.P.

Date: September 27, 2001

By: 

William H. Wright

Registration No. 36,312

Attorney for Applicant(s)

500 South Grand Avenue, Suite 1900
Los Angeles, California 90071
Telephone: 213-337-6700
Facsimile: 213-337-6701

THIS PAGE BLANK (USPTO)



BREVET D'INVENTION

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le

26 JUIN 2001

Pour le Directeur général de l'Institut
national de la propriété industrielle
Le Chef du Département des brevets

Martine PLANCHE

INSTITUT
NATIONAL DE
LA PROPRIÉTÉ
INDUSTRIELLE

SIEGE
26 bis, rue de Saint Petersburg
75800 PARIS cedex 08
Téléphone : 33 (1) 53 04 53 04
Télécopie : 33 (1) 42 93 59 30
www.inpi.fr

THIS PAGE BLANK (USPTO)



26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : 01 53 04 53 04 Télécopie : 01 42 94 86 54

BREVET D'INVENTION

CERTIFICAT D'UTILITÉ

Code de la propriété intellectuelle - Livre VI



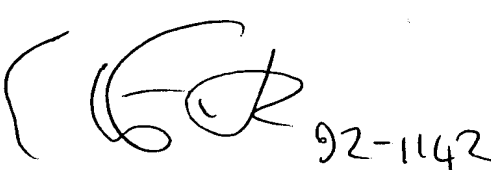
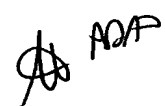
N° 11354*01

REQUÊTE EN DÉLIVRANCE 1/2

Cet imprimé est à remplir lisiblement à l'encre noire

DB 540 W / 260899

REMISE DES PIÈCES DATE 12 JUL 2000 LIEU 75 INPI PARIS N° D'ENREGISTREMENT 0009105 NATIONAL ATTRIBUÉ PAR L'INPI DATE DE DÉPÔT ATTRIBUÉE PAR L'INPI 12 JUL 2000		1 NOM ET ADRESSE DU DEMANDEUR OU DU MANDATAIRE À QUI LA CORRESPONDANCE DOIT ÊTRE ADRESSÉE Cabinet REGIMBEAU 26, avenue Kléber 75116 PARIS FRANCE	
Vos références pour ce dossier (facultatif) 238650 d18980 ELF			
Confirmation d'un dépôt par télécopie <input type="checkbox"/> N° attribué par l'INPI à la télécopie			
2 NATURE DE LA DEMANDE	Cochez l'une des 4 cases suivantes		
Demande de brevet	<input checked="" type="checkbox"/>		
Demande de certificat d'utilité	<input type="checkbox"/>		
Demande divisionnaire	<input type="checkbox"/>		
<i>Demande de brevet initiale</i>	N°	Date	/ /
<i>ou demande de certificat d'utilité initiale</i>	N°	Date	/ /
Transformation d'une demande de brevet européen <i>Demande de brevet initiale</i>	<input type="checkbox"/>	N°	Date / /
3 TITRE DE L'INVENTION (200 caractères ou espaces maximum) SYSTEME DE CONVERSION DE DOCUMENTS A STRUCTURE ARBORESCENTE PAR PARCOURS SELECTIF DE LADITE STRUCTURE			
4 DÉCLARATION DE PRIORITÉ OU REQUÊTE DU BÉNÉFICE DE LA DATE DE DÉPÔT D'UNE DEMANDE ANTÉRIEURE FRANÇAISE	Pays ou organisation Date / / N° Pays ou organisation Date / / N° Pays ou organisation Date / / N° <input type="checkbox"/> S'il y a d'autres priorités, cochez la case et utilisez l'imprimé «Suite»		
5 DEMANDEUR	<input type="checkbox"/> S'il y a d'autres demandeurs, cochez la case et utilisez l'imprimé «Suite»		
Nom ou dénomination sociale	JAXO EUROPE		
Prénoms			
Forme juridique	SOCIETE A RESPONSABILITE LIMITEE		
N° SIREN	421448879		
Code APE-NAF			
Adresse	Rue	705, rue St Hilaire, 34000 MONTPELLIER	
	Code postal et ville		
Pays	FRANCE		
Nationalité	Française		
N° de téléphone (facultatif)			
N° de télécopie (facultatif)			
Adresse électronique (facultatif)			

REMISE DES PIÈCES DATE 12 JUIL 2000 LIEU 75 INPI PARIS N° D'ENREGISTREMENT NATIONAL ATTRIBUÉ PAR L'INPI 0009105		Réservé à l'INPI	DB 540 W / 260899
V s références pour ce dossier : <i>(facultatif)</i>		238650 d18980 ELF	
6 MANDATAIRE			
Nom			
Prénom			
Cabinet ou Société		Cabinet REGIMBEAU	
N °de pouvoir permanent et/ou de lien contractuel			
Adresse	Rue	26, avenue Kléber	
	Code postal et ville	75116 PARIS	
N° de téléphone <i>(facultatif)</i>		01 45 00 92 02	
N° de télécopie <i>(facultatif)</i>		01 45 00 46 12	
Adresse électronique <i>(facultatif)</i>		info@regimbeau.fr	
7 INVENTEUR (S)			
Les inventeurs sont les demandeurs		<input type="checkbox"/> Oui <input checked="" type="checkbox"/> Non Dans ce cas fournir une désignation d'inventeur(s) séparée	
8 RAPPORT DE RECHERCHE		Uniquement pour une demande de brevet (y compris division et transformation)	
Établissement immédiat ou établissement différé		<input checked="" type="checkbox"/> <input type="checkbox"/>	
Paiement échelonné de la redevance		Paiement en deux versements, uniquement pour les personnes physiques <input type="checkbox"/> Oui <input type="checkbox"/> Non	
9 RÉDUCTION DU TAUX DES REDEVANCES		Uniquement pour les personnes physiques <input type="checkbox"/> Requête pour la première fois pour cette invention <i>(joindre un avis de non-imposition)</i> <input type="checkbox"/> Requête antérieurement à ce dépôt <i>(joindre une copie de la décision d'admission pour cette invention ou indiquer sa référence) :</i>	
Si vous avez utilisé l'imprimé «Suite», indiquez le nombre de pages jointes			
10 SIGNATURE DU DEMANDEUR OU DU MANDATAIRE (Nom et qualité du signataire)		VISA DE LA PRÉFECTURE OU DE L'INPI	
			

DÉPARTEMENT DES BREVETS

26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08

Téléphone : 01 53 04 53 04 Télécopie : 01 42 94 86 54

DÉSIGNATION D'INVENTEUR(S) Page N° . 1 . / . 1 .

(Si le demandeur n'est pas l'inventeur ou l'unique inventeur)

Cet imprimé est à remplir lisiblement à l'encre noire

DB 113 W / 260899

Vos références pour ce dossier (facultatif) 238650 d18980 ELF			
N° D'ENREGISTREMENT NATIONAL		0009105	
TITRE DE L'INVENTION (200 caractères ou espaces maximum)			
SYSTEME DE CONVERSION DE DOCUMENTS A STRUCTURE ARBORESCENTE PAR PARCOURS SELECTIF DE LADITE STRUCTURE			
LE(S) DEMANDEUR(S) :			
JAXO EUROPE : 705, rue St Hilaire, 34000 MONTPELLIER - FRANCE			
DESIGNE(NT) EN TANT QU'INVENTEUR(S) : (Indiquez en haut à droite «Page N° 1/1» S'il y a plus de trois inventeurs, utilisez un formulaire identique et numérotez chaque page en indiquant le nombre total de pages).			
Nom		RICHARD Pierre	
Prénoms			
Adresse	Rue	5, rue Joseph Vidal 34000 MONTPELLIER FR	
	Code postal et ville		
Société d'appartenance (facultatif)			
Nom			
Prénoms			
Adresse	Rue		
	Code postal et ville		
Société d'appartenance (facultatif)			
Nom			
Prénoms			
Adresse	Rue		
	Code postal et ville		
Société d'appartenance (facultatif)			
DATE ET SIGNATURE(S) DU (DES) DEMANDEUR(S) OU DU MANDATAIRE (Nom et qualité du signataire)			

DOCUMENT COMPORTANT DES MODIFICATIONS

PAGE(S) DE LA DESCRIPTION OU DES REVENDECATIONS OU PLANCHE(S) DE DESSIN			R.M.*	DATE DE LA CORRESPONDANCE	TAMPON DATEUR DU CORRECTEUR
Modifiée(s)	Supprimée(s)	Ajoutée(s)			
Devis 1 à 28				16/11/00	22 NOV. 2000 - B B

Un changement apporté à la rédaction des revendications d'origine, sauf si celui-ci découle des dispositions de l'article R.612-36 du code de la Propriété Intellectuelle, est signalé par la mention « R.M. » (revendications modifiées).

La présente invention concerne d'une façon générale un système et un procédé de conversion dynamique et paramétrable d'un ou plusieurs flux de données balisées.

Arrière-plan de l'invention

5

1. L'Internet et les téléphones portables

10 En matière de communication et d'échanges d'informations, les années 1990 ont témoigné d'innovations technologiques dont les répercussions ont aujourd'hui une étendue mondiale. Deux grandes causes paraissent être à la base de ce phénomène : la montée en puissance de l'Internet d'une part, et la multiplication des appareils portables « sans fil » -- téléphones, assistants électroniques -- d'autre part.

15 On peut différencier ces deux causes car elles n'ont eu au départ ni les mêmes origines, ni les mêmes cibles commerciales. L'Internet ressort typiquement du domaine de l'informatique : la population concernée connaît le maniement d'un ordinateur et de ses programmes. Le domaine du « sans fil », quant à lui, appartient au monde de la téléphonie. Il est destiné à une clientèle grand public.

20 Bien que différentes au départ dans leur conception, leurs méthodes propriétaires et leurs applications, il apparaît aujourd'hui que ces deux techniques de communication électronique vont se rapprocher jusqu'à se confondre. Les consortiums de constructeurs (W3C , WAPforum), les accords réciproques entre les grands industriels de la téléphonie et de l'informatique sont aujourd'hui les premiers jalons
25 de cette unification.

Un objectif essentiel de la présente invention est de permettre de produire des données exploitables par les téléphones portables accédant au réseau Internet.

2. Les langages balisés

Le format des données circulant sur l'Internet obéit à des normes précises et internationales qui ont favorisé son essor. Cette normalisation s'applique à toutes les couches participant à l'échange de ces données : la couche de transport TCP/IP (pour « Transmission Control Protocol/Internet Protocol » en terminologie anglo-saxonne), le protocole de requête/réponse HTTP (pour « HyperText Markup Language » en terminologie anglo-saxonne), et finalement le contenu de la réponse elle-même, la page de la Toile (« Web » en terminologie anglo-saxonne). Le langage utilisé pour décrire une page de la Toile est de type « balisé » : il comporte ainsi des balises, c'est-à-dire des suites de caractères qui se différencient du texte parce qu'elles sont encadrées par des caractères spéciaux « < » et « > ». La pièce de logiciel qui décode le document balisé s'appelle un analyseur syntaxique (« Parser » en terminologie anglo-saxonne).

15

Les balises créent une véritable hiérarchie au sein du texte, indiquant, par exemple, qu'il s'agit du titre du 2ème sous-chapitre du 3ème chapitre du corps du document. C'est par l'analyse de cette hiérarchie que le navigateur de la Toile décide notamment du rendu typographique de chaque portion de texte : titres, paragraphes, tableaux, images.

20

Outre la syntaxe de l'écriture des balises qui permet de les distinguer du texte normal, la hiérarchie des balises fait aussi l'objet de règles : par exemple, un paragraphe ne contient jamais d'autres paragraphes, mais un tableau peut contenir d'autres tableaux. Publiée en 1986, la norme SGML (pour « Standard Generalized Markup Language » en terminologie anglo-saxonne) est la première qui définit la formalisation de ces règles, en particulier en termes de syntaxe d'écriture et de description des contraintes hiérarchiques. Cette norme SGML donne du langage balisé une description toutefois purement formelle, sans préciser quelles balises une application devra utiliser. La norme SGML est « générique », au sens qu'elle définit les règles communes aux langages de balisage. Mais elle n'est pas, en elle-même, un

25

30

langage de balisage contrairement à HTML, qui est le langage standard de balisage des pages de la Toile.

3. Médiocrité du balisage des pages de la Toile

5

Le langage HTML impose qu'une page contienne une suite de balises fixes bien précises (typiquement une centaine ou davantage), les règles du balisage se conformant en théorie aux principes édictés par la norme SGML. Mais dans la pratique, et ce pour des raisons de précedence historique et de complexité de la norme SGML, la majorité des informations existantes sur la Toile entrent en conflit avec les règles prescrites.

10

C'est notamment l'absence d'outils de validation qui a permis la prolifération de ces milliards de pages non conformes à la norme SGML, sachant que les concepteurs de ces pages pouvaient librement recourir à des truquages pour arriver à leur objectif premier, à savoir la qualité du rendu typographique obtenu lors de la visualisation des informations. Par exemple, un titre de chapitre au milieu d'un paragraphe permet d'obtenir des caractères gras dans une police de taille plus importante, alors qu'un chapitre au sens de la norme ne peut pas se concevoir comme étant imbriqué dans un paragraphe. Au sens du balisage, c'est une erreur grave, mais les logiciels de navigation, qui ne sont pas des outils de validation, ne le signalent pas.

15

20

Il en résulte qu'aujourd'hui, la majorité des données existantes sur les sites Internet sont vouées à une application unique spécifique qui est leur visualisation sur écran d'ordinateur par le biais des logiciels de navigation standard tels que « Netscape Navigator » ou « Internet Explorer » (marques déposées au nom de leurs titulaires respectifs).

25

Mais la réutilisation de ces pages en vue d'autres applications (comme par exemple l'affichage sur un écran de téléphone portable) ne serait pas envisageable autrement que par une dégradation conséquente du contenu, sans garantie de résultat fiable.

30

Un premier objet de la présente invention est de permettre l'adaptation des données actuellement existantes sur les sites en vue de leur traitement par une diversité d'applications, sans nécessiter une refonte préalable de ces données.

5

4. La norme XML

Des outils de validation sont apparus à peu près dans les années 1993, en révélant au passage l'étendue des erreurs du balisage de la plupart des pages de la Toile. A cet
10 égard, les organisations qui régissent la Toile se livrent à de nombreuses tentatives de normalisation : définition de règles hiérarchiques strictes d'une part, mais aussi laxistes d'autre part, dites transitionnelles, dans le but déclaré de récupérer le plus grand nombre de documents erronés. Malgré ce, il devient impossible d'endiguer le flot croissant de documents générés en violation de la norme. Les mauvaises
15 habitudes sont déjà ancrées et les auteurs refusent ces changements qu'ils ne comprennent pas.

Pour prendre en main le problème, un consortium dénommé W3C a été créé et, dès 1997, a publié la norme XML (pour « eXtensible Markup Language » en
20 terminologie anglo-saxonne) qui dérive de la norme SGML en la simplifiant, mais qui aussi renforce de façon stricte la syntaxe. La norme XML a remporté un succès considérable dans le monde industriel d'abord, et se popularise peu à peu. Parmi les nombreux exemples d'applications, les téléphones portables accédant à l'Internet utilisent le langage de balisage WML qui respecte scrupuleusement la norme XML ;
25 dans le même esprit, les constructeurs de base de données fournissent les moyens d'extraction du contenu dans un format XML ; ainsi la spécification EDI (Electronic Data Interchange en terminologie anglo-saxonne) est en voie de normalisation XML.

Il apparaît toutefois pour les praticiens que la norme XML, avec ses règles strictes,
30 n'enrichit pas, voire appauvrit la norme SGML, plus permissive sur les omissions de balises courantes.

Comme on le verra plus loin, un autre objectif de la présente invention est la mise au point d'un analyseur syntaxique SGML validant à tolérance de fautes. Il comporte des algorithmes complexes résolvant la plupart des anomalies de balisage et produit un flux de données en conformité stricte avec la norme XML.

5. Conversion

Comme vu plus haut, le respect des règles de hiérarchie des balises permet la production de documents accessibles à une multiplicité d'applications diverses. En effet, le but du balisage est d'informer du rôle de telle ou telle portion d'information (de texte, d'image), sans présumer de l'usage qui doit en être fait. Chaque application spécifique traitant ces documents procède d'une manière qui lui est propre.

Dérivant de la norme SGML, la norme XML est « générique », en ce sens qu'il n'est pas un langage de balisage par lui même. Une page de la Toile qui respecterait la norme XML est écrite dans le langage XHTML (pour « eXtended HyperText Markup Language » en terminologie anglo-saxonne). A supposer qu'une telle page existe, une question qui se pose est de savoir comment traiter les informations qui y sont contenues afin de les présenter sur un téléphone mobile qui présente des caractéristiques fondamentalement différentes de celles d'un ordinateur personnel.

Sur ce sujet, les publications d'un groupe de travail dénommé WAP Forum (WAP pour « Wireless Application Protocol » en terminologie anglosaxonne) contiennent des recommandations qui établissent que, pour produire un document pouvant être visualisé sur un téléphone cellulaire (en particulier selon la norme WML pour « Wireless Markup Language » en terminologie anglo-saxonne), il faut d'abord construire un « super document » au format XML. Ce document, traité par une transformation de type XSL (pour « eXtensible Style Language » en terminologie anglo-saxonne) pourra générer soit du code en HTML , soit du code en WML, selon

le script (à savoir une feuille de style au format XSL) utilisé par la transformation. Ceci est illustré schématiquement sur la figure 1 des dessins.

- 5 Il découle de ces évolutions normatives qu'il n'est pas question de pouvoir réutiliser les documents HTML existants, mais que de nouveaux documents doivent être créés de toutes pièces de telle manière qu'une transformation appropriée par un script XSL permette de reconstituer le document dans sa version HTML, si nécessaire.

- 10 Mais à ce sujet, on peut s'attendre à ce que la plupart des sites de la Toile refuseront l'approche drastique proposée par le WAP Forum. En particulier, les sites « personnels » ne voudront pas changer par manque de temps pour la ré-écriture. Quant aux sites « professionnels moyens » (journaux, bourse, transports, etc.), leurs producteurs ont déjà beaucoup investi dans l'écriture HTML de leurs pages, dans la création des automatismes dans les langages cgi-bin et JavaScript (sachant que
- 15 l'environnement JavaScript que procure XML n'est pas compatible). Enfin et surtout, les producteurs de ces sites ont lourdement investi dans la mise au point du mécanisme d'ensemble, notamment pour faire face à l'incompatibilité des navigateurs Netscape / Internet Explorer.

- 20 En outre, la norme XML relève d'une « culture informatique » différente : l'histoire a révélé les difficultés du langage SGML qui ne s'est jamais vraiment imposé principalement parce que le concept de hiérarchie du balisage n'est ni simple, ni populaire. Certes le langage XML a simplifié la syntaxe du langage SGML, mais, comme avec SGML, un document écrit dans le langage XML se doit de respecter la
- 25 hiérarchie des balises. Ainsi, en dépit de la popularité croissante de XML, qui a été bénéfique pour la diffusion du concept de balisage, on ne doit pas s'attendre à ce que le langage XML apporte beaucoup plus que le langage SGML, son ancêtre.

- Il ne semble donc pas réaliste de s'attendre à une généralisation massive de
- 30 l'approche XML/XSL, si ce n'est peut-être à très long terme, mais l'on ignore ce que sera devenu alors cette technologie.

Résumé de l'invention

La présente invention a en particulier pour objet d'offrir une alternative à cette
 5 évolution normative, qui permette, contrairement aux recommandations qui imposent
 une ré-écriture des pages de façon compatible XML, de pouvoir effectuer des
 transformations sur la plus grande partie des pages HTML existantes, y compris la
 grande proportion de celles-ci qui comportent des fautes d'écriture.

10 Plus précisément, grâce à un mécanisme à conversions multiples extrêmement
 souple, la présente invention vise à ce que les données à convertir ne nécessitent pas
 d'être normalisées en XML, ni même de respecter strictement la hiérarchie des
 balises. Ceci est illustré schématiquement sur la figure 2 des dessins.

15 Un autre objet encore de la présente invention est d'offrir, pour réaliser de telles
 conversions, une syntaxe d'écriture d'instructions particulièrement simple à mettre
 en œuvre.

Ainsi la présente invention propose système de conversion de documents, destiné à
 20 transformer un premier document existant dans un premier format balisé à structure
 arborescente de nœuds en un second document dans un second format balisé à
 structure arborescente de noeuds, caractérisé en ce qu'il comprend :

un ensemble de procédures-gabarits (Template), chaque procédure-gabarit
 dudit ensemble contenant une première information de sélection d'une étendue de
 25 parcours de la structure arborescente du premier document, une seconde information
 de sélection d'un type de balise du premier document auquel la procédure gabarit
 peut s'appliquer, et un ensemble d'une ou plusieurs actions, au moins certaines
 procédures gabarits contenant des actions de collecte d'informations délimitées par
 les balises dont le type correspond à celui défini par la seconde information de
 30 sélection de ces procédures gabarits,

un programme contenant des instructions standard (applyTemplates) pour sélectivement appeler lesdites procédures gabarits,

- des moyens de traitement répondant audit programme et aptes, lors de l'appel d'une procédure-gabarit, à parcourir les nœuds de la structure arborescente
 5 inclus dans son étendue telle que définie par la première information de sélection de ladite procédure-gabarit, et pour chaque nœud parcouru, à déterminer si le type de balise correspondant audit nœud correspond à la seconde information de sélection, et, dans l'affirmative, à exercer la ou les actions correspondantes, et
 10 des moyens aptes, à partir des actions de collecte contenues dans les procédures-gabarits exécutées, pour construire le second document.

Certains aspects préférés, mais non limitatifs, du système selon l'invention sont les suivants :

- 15 - les actions d'au moins certaines procédures-gabarits contiennent des instructions d'appel d'autres procédures gabarits.
- la première information de sélection d'étendue d'une procédure-gabarit est basée sur un nœud de départ constitué par le nœud à partir duquel ladite procédure-gabarit
 20 est appelée.
- au moins certaines procédures-gabarits contiennent des actions de création de variables temporaires cloîtrées, tandis que chacune de ces variables cloîtrées est héritée par toute procédure-gabarit appelée par de telles procédures-gabarits.
- 25 - les procédures-gabarits comprennent une procédure-gabarit de base possédant la seconde information de sélection correspondant à un nœud racine de la structure arborescente du premier document, tandis que les actions de ladite procédure-gabarit de base comprennent un appel de procédures-gabarits.

- les actions de ladite procédure-gabarit de base comprennent un appel de procédures-gabarits dont la seconde information de sélection correspond à un type de balise « corps ».
- 5 - la structure arborescente de nœuds du premier document comprend des nœuds de type « cadre », et en ce que les actions de ladite procédure-gabarit de base comprennent un appel de procédures-gabarits dont la seconde information de sélection correspond à un type de balise « ensemble de cadres ».
- 10 - au moins certaines procédures-gabarits comprennent au moins une action de redirection vers un premier document différent, sur lequel lesdits moyens de traitement sont appliqués sur la base du même programme.
- 15 - au moins certaines procédures gabarits comprennent au moins une action conditionnelle basée sur le contenu d'une adresse d'accès au premier document.
- 20 - au moins certaines procédures-gabarits comprennent au moins une action constituant une méthode d'un objet et/ou au moins une action constituant une fonction locale programmée.
- 25 - les moyens de construction du second document comprennent des actions d'écriture contenues dans certaines procédures-gabarits dont la seconde information de sélection définit une balise à contenu, lesdites actions d'écriture étant aptes à assembler de façon prédéterminée au moins une partie des contenus desdites balises.
- 30 - les premiers documents comprennent des pages structurées dans un langage balisé standard adapté à une consultation sur poste informatique client via l'Internet, tandis

que les seconds documents comprennent des pages structurées dans un second langage balisé standard adapté à une consultation sur appareil miniature portable.

- le système constitue un gradin d'une architecture multi-gradins.

5

L'invention propose en outre une architecture informatique multi-gradins, caractérisée en ce qu'elle comprend en succession, reliés par réseau informatique, un premier gradin de consultation de données sur poste client, un second gradin d'application sur serveur, un troisième gradin d'agrégation de données comprenant
10 un système de conversion tel que défini ci-dessus et un quatrième gradin comprenant une pluralité de types de sources de données indépendantes.

15

Par exemple, dans une application de consultation notamment via l'Internet de données agglomérées provenant de sources différentes, cette architecture est en outre caractérisée en ce que la pluralité de types de sources de données indépendantes comprennent au moins deux types de sources parmi les serveurs Internet, les serveurs d'annuaires à accès rapide et les serveurs de bases de données à accès standard par requêtes.

20

L'invention propose enfin une architecture informatique/téléphonique multigradins, caractérisée en ce qu'elle comprend en succession, reliés par réseau informatique et par réseau téléphonique sans fil, un premier gradin de consultation de données sur un appareil portable tel qu'un téléphone portable, un second gradin de transport de données sans fil, un troisième gradin comprenant un système de conversion de
25 documents tel que défini plus haut et un quatrième gradin comprenant au moins un type de source de données.

30

Par exemple, dans une application de consultation de données de serveurs Internet standard à partir de téléphones portables, cette architecture est en outre caractérisée en ce que ledit type de source de données consiste en des pages structurées dans un langage balisé standard adapté à une consultation sur poste informatique client,

lesdites pages constituant des premiers documents, et en ce que le système de conversion compris dans le troisième gradin est apte à construire des seconds documents dans un second langage balisé standard adapté à une consultation sur appareil miniature portable.

5

Brève description des dessins

D'autres aspects, buts et avantages de la présente invention apparaîtront mieux à la lecture de la description détaillée suivante d'une forme de réalisation préférée de celle-ci, donnée à titre d'exemple non limitatif et faite en référence aux dessins annexés, sur lesquels :

10

la figure 1, déjà décrite, est une représentation schématique de l'approche classique pour adapter un même contenu à différents environnements physiques et logiques de navigation sur la Toile,

15

la figure 2, également déjà décrite, est une représentation schématique de l'approche retenue selon la présente invention pour atteindre le même type d'objectif,

20

la figure 3 illustre schématiquement un ensemble de blocs fonctionnels mis en œuvre selon la présente invention,

25

les figures 4 et 5 illustrent schématiquement deux exemples d'architecture à gradins dans lesquelles peut s'intégrer le système de la présente invention,

la figure 6 illustre schématiquement une transformation d'arbre en flux réalisée selon l'invention,

30

la figure 7 illustre plus en détail une transformation d'arbre selon l'invention,

les figures 8 à 15, 17 et 18 illustrent des extraits de scripts écrits dans le langage ECMAScript et utilisés dans un exemple de mise en œuvre de la présente invention,

la figure 16 illustre une structure d'arbre d'une partie d'un document de départ
5 concernant l'exemple de mise en œuvre précité,

la figure 19 illustre la présentation visuelle d'un document d'origine et d'un document converti selon la présente invention,

10 les figures 20 à 27 illustrent différentes parties d'un journal de trace susceptible d'être généré par le système de la présente invention, et

la figure 28 illustre une Description Technique de Document définissant le format des scripts de conversion utilisés par la présente invention.

15

Description détaillée d'une forme de réalisation préférée de l'invention

On notera à titre préliminaire que la plupart des concepts, spécifications, normes, etc. auxquels se réfère la présente description font l'objet de descriptions détaillées
20 accessibles sur le site Internet www.w3.org du Consortium W3C précité.

On notera également que les différents noms de marques cités dans la présente description et apparaissant sur les dessins sont des marques déposées aux noms de leurs titulaires respectifs.

25

1. L'arbre d'un document

Comme on le verra en détail plus loin, la conversion que vise à réaliser la présente invention nécessite non seulement d'examiner le contenu des informations, mais
30 également de prendre en compte leur signification dans un contexte : données d'un

paragraphe, d'un lien hypertexte, d'une image, d'un son, numéro de carte de crédit, etc.

- 5 Le mécanisme de conversion selon l'invention s'appuie sur la possibilité de représentation en arbre d'un document, propriété fondamentale de tout document correctement balisé : ainsi la branche « corps du document » d'un document possède des sous-branches « chapitre » qui possèdent à leur tour des branches « sous-chapitre », etc.
- 10 La théorie des arbres (navigation et transformations) est une des théories fondamentales du traitement de données. C'est ainsi que la conversion du contenu en fonction du contexte utilisée selon la présente invention met en œuvre une technique de transformation d'arbre connue en soi.
- 15 La théorie des arbres étant abstraite, son application concrète nécessite de décrire :
- les caractéristiques de l'arbre (que sont les branches, que sont les feuilles ?) ;
 - la navigation au sein de l'arbre (comment par exemple retrouver un ancêtre dans un arbre généalogique ?) ; et enfin
 - la transformation de l'arbre (comment par exemple créer un arbre de cousins
- 20 germains ?).

- 25 Le consortium W3C précité préconise l'utilisation de la spécification DOM (« Document Object Model » en terminologie anglo-saxonne) pour décrire l'arbre XML en termes de modèle de document, et préconise l'utilisation des langages XSL et XSLT (pour « eXtensible Style Language Transformations » en terminologie anglo-saxonne) pour leur transformation.

La mise au point de la présente invention a conduit à une étude approfondie de ces méthodes, en les remplaçant dans un contexte industriel d'utilisation.

A cet égard, un objectif annexe de la présente invention a été d'optimiser les standards existants.

5 Prônée par les techniques de script dynamique de type DHTML (pour « Dynamic HTML » en terminologie anglo-saxonne), la spécification DOM s'impose comme le moyen reconnu qui permet la navigation dans les nœuds de l'arbre d'un document.

10 Ainsi la présente invention utilise la syntaxe DOM pour décrire l'arbre du document, avec certaines extensions de type agent de correspondance MATCHER, que l'on détaillera plus loin, destinées à obtenir une plus grande efficacité dans les traversées de l'arbre.

15 On observera ici que la transformation édictée par le langage XSL et par les transformations XSLT se fonde sur une approche classique « sélection de nœuds/choix de gabarits ». Cette approche, déjà préconisée par la norme DSSSL (pour « Document Style Semantic and Specification Language » en terminologie anglo-saxonne), semble être l'un des points positifs du langage XSL. Mais le langage XSL est, dans son application industrielle, relativement complexe, et donc met en jeu des connaissances spécifiques et nécessite l'apprentissage d'un nouveau langage de
20 programmation. En outre, XSL/XSLT ne permet pas la navigation DOM et ne permet pas de bénéficier des outils de programmation les plus reconnus.

25 En d'autres termes, la présente invention telle qu'on va la décrire en détail constitue un choix optimal parmi une multiplicité d'approches conflictuelles, en proposant une technique d'assemblage unique harmonisant les interactions entre les outils les plus pratiqués dans le monde de l'industrie documentaire.

2. La conversion de documents

30 La conversion d'un document balisé selon la présente invention est paramétrée par un script, dit « script de conversion », qui est programmé dans un langage de

conversion essentiel pour la mise en œuvre de la présente invention. Ce langage « fédérateur » centralise la communication avec les divers modules fonctionnels indépendants requis pour la conversion :

- * module DOM d'accès aux nœuds de l'arbre du document en entrée, module
- 5 d'analyse et de recherche de texte (incluant un moteur de recherche par expression régulière) ;
- * module d'écriture du document résultat (incluant un moteur d'épuration);
- * module d'itération (traversée de l'arbre par sélection de nœuds) ;
- * module de maintenance des variables (au niveau global, ou dans l'environnement
- 10 de service) ;
- * module de création d'objets Java ;
- * module de trace (debogage).

La syntaxe du langage de conversion est dans le présent exemple ECMAScript , tel

15 que strictement spécifié par la norme du même nom. ECMAScript – dont JavaScript est une extension – est un langage simple, couramment pratiqué par les programmeurs de la Toile. Comme on l'a indiqué plus haut, ceci évite de créer pour le programmeur une nouvelle syntaxe, tout en évitant de passer par les scripts balisés selon le langage XSL. Pour tous compléments d'informations sur le langage

20 ECMAScript, on pourra notamment se référer aux informations disponibles sur le site Internet www.ecma.ch.

Les détails du fonctionnement du langage de conversion sont donnés plus loin.

25 3. Description fonctionnelle de l'invention

Le système et le procédé de la présente invention combinent trois fonctions essentielles, dont la première est toutefois facultative pour des documents balisés convenablement construits :

a) regroupement et transformation des documents balisés qui dérogent aux standards

5 Comme on l'a dit plus haut, dans les milliards d'octets de pages de données électroniques existant sur la Toile, la majorité de ces pages sont en violation de l'aspect structuration de la norme HTML et, en conséquence, ne peuvent être traitées ; cette fonction a pour but de récupérer la plupart des données existantes et de les normaliser : elle met en œuvre des algorithmes à tolérance de fautes développés de manière à restituer un « arbre de document » conforme aux prescriptions de XML.

10

b) langage de conversion de cette transformation destiné à des applications en environnement industriel.

15 Ce langage de conversion a été conçu pour offrir une souplesse remarquable dans le paramétrage de la conversion de tout document balisé. Les caractéristiques de ce langage de conversion en font un langage puissant , tout en restant facile à mettre en œuvre.

20 *c) outil convertisseur XGate intégrant en un seul produit la transformation en mode flux dynamique*

25 Cet outil corrige les documents à convertir, interprète le script de conversion et produit le résultat dans un mode de « flux continu » : un nombre théoriquement illimité de documents peuvent être convertis en simultané, sous réserve de la bande passante liée notamment à la puissance de calcul et à la mémoire disponible.

30 La bibliothèque des programmes de conversion est écrite de préférence en langage Java. Une application particulièrement révélatrice consiste en la traduction automatique du contenu de sites quelconques de la Toile. Elle utilise un script approprié, écrit dans le langage de conversion, pour traiter le plus grand nombre de sites « en aveugle ».

D'autres scripts écrits dans le langage de conversion peuvent être développés pour la traduction sur les téléphones portables de sites HTML ciblés (par exemples aéroports, résultats sportifs, etc.).

5

Les bases techniques utilisées pour la mise en œuvre de la présente invention sont de préférence les suivantes :

- * Langage Java (Java 1.2, 100% Pure Java), interpréteur ECMAScript ;
- * Architecture à gradins pour le contrôle des demandes à cibles multiples, le regroupement et l'organisation en document balisés des réponses ;
- * Analyseur syntaxique SGML pour une analyse à tolérance de fautes des documents balisés, pour la mise aux normes XML, et pour la génération de l'arborescence du document résultant en tant que mode dynamique de représentation du contenu des données originelles ;
- * Transformation dynamique d'arbre à arbre via un script de type : « modèle/correspondance/sélection » (« template/match/select » en terminologie anglo-saxonne), mais introduisant d'autres concepts uniques (interpréteur ECMAScript, recherche par expressions régulières, accès direct aux nœuds par navigation DOM, environnement de transformation et de service).

20

Les applications de la présente invention sont nombreuses :

- * Commerce électronique ;
- * Conversion de page de la Toile à destination de matériels sans fil ;
- * Suivi de concurrence par analyse intelligente de contenus ;
- * Génération de flux multimédia à partir de sources multiples (musique, images, etc.).

30

Les avantages qu'elle apporte sont principalement les suivants :

- * Sécurité et facilité de déploiement par la centralisation de l'accès aux ressources ;

- * Facilité de mise en œuvre et d'adaptation des scripts de transformation ;
- * Extensibilité et efficacité du fait de la séparation des couches d'accès et de logique applicative (transformation) ;
- * Stabilité : l'architecture "ouverte" du système met à profit les techniques standard utilisées sur l'Internet : HTTP en tant que protocole de transfert, XML en tant que format universel de données structurées, ECMAScript en tant que langage de transformation.

4. Description détaillée

4.1. Ensemble des modules fonctionnels du convertisseur XGate

Le schéma illustré sur la figure 3 des dessins représente les composants essentiels du convertisseur.

Les données sources sont symbolisées par le module « Back-ends ».

Le module « Business Applications » représente la logique de l'application cliente, alimentée par les données transformées par le convertisseur XGate. Ce module constitue par exemple la partie logique applicative d'une l'application de commerce électronique, telle qu'elle est présentée dans l'exemple donné plus loin. Comme indiqué dans ce même exemple, la Business Application communique par les ports HTTP, en constituant un gradin spécifique au niveau TCP/IP. Mais dans les cas les plus simples le module « Business Application » peut ne pas exister. Le client communique alors directement avec la sortie du convertisseur XGate, l'outil de communication étant le plus souvent un navigateur standard de la Toile.

Le rôle du module « Broker » est de décomposer chaque requête en ordres à destination d'un module de normalisation « Normalizer » et d'un module de transformation « Transformer ». Le module « Broker » a accès à un module « Repository » destiné à enregistrer les requêtes les plus courantes et les profils qui y

sont associés. Par exemple, dans le cas d'une transformation d'informations codées en HTML vers des informations codées en WML (typiquement pour rendre accessibles sur des téléphones portables des informations accessibles sur des sites Internet), le module « Repository » connaît les caractéristiques physiques du modèle de téléphone portable qui soumet la requête (dimensions de l'écran, etc.) Il connaît
 5 davantageusement aussi le profil de l'appelant (ses sites préférés, etc.).

Le module « Normalizer » répond à une requête (flèche « Actions ») en activant le nombre de données sources nécessaires. Le module « Normalizer » restitue ces
 10 données au format XML. C'est dans ce module « Normalizer » que se trouve le composant d'analyse syntaxique à tolérance de fautes. Mais ce module peut également contenir d'autres composants : par exemple, si les données d'un « Back-ends » auquel il accède sont structurées en une base de données, le module « Normalizer » doit pouvoir engendrer un balisage spécifique à ce type de requête :
 15 c'est le rôle du composant de filtrage « Filter » de engendrer ce balisage.

On notera que, pour une requête « Actions » donnée, le nombre de documents XML générés dépend du nombre de sources de données activées.

20 Le module « Transformer » répond à une requête (flèche de type « Layout ») en lisant le flux XML émis par le module « Normalizer » et en lui appliquant le ou les scripts de transformation tels que choisis par le module « Broker ». Si besoin est, le module « Transformer » a la possibilité de retourner une requête complémentaire au module « Broker » (ce qui n'est pas symbolisé sur la figure 3 pour des raisons de
 25 lisibilité).

4.2. Exemples d'architectures fonctionnelles

4.2.1. Application de commerce électronique

La figure 4 illustre l'intégration du convertisseur XGate de la présente invention au sein d'une architecture à 4 gradins pour une application de commerce électronique.

Dans le présent exemple, le but de cette application est de fournir aux clients un catalogue de produits comportant images, prix, et adresses des fournisseurs. Les clients utilisent un navigateur de Toile pour la visualisation des résultats.

Les différentes informations nécessaires proviennent de source hétérogènes, ici une base de données « SQL Server » pour les prix, un serveur « LDAP Server » pour les adresses (Lightweight Directory Access Protocol en terminologie anglo-saxonne - il s'agit d'un protocole normalisé facilitant la recherche d'informations organisées en annuaire ou en répertoire, telle que la recherche de personnes classées selon leur nom, leur entreprise, leur pays, etc.), et un serveur « Web Server » pour des pages de la Toile décrivant les produits (texte, images, son, etc.).

15

Une logique applicative bien conçue se doit d'être affranchie de toute question relative à l'obtention des informations qu'elle traite. De même, la façon dont ces informations sont affichées physiquement sur l'écran du demandeur n'est pas du ressort de l'application. Pour ce dernier point, le navigateur de Toile traduira le flux HTML émis par la logique applicative en termes d'instructions d'affichage qui produiront le résultat correspondant sur l'écran de l'ordinateur du client.

20

Le résultat produit par la logique applicative est donc un flux HTML dont l'interprétation en images est effectuée sur l'ordinateur du client : c'est la notion de "gradin" indiquée plus haut.

25

Chaque gradin possède une responsabilité bien définie, et l'interface de présentation au niveau du poste client constitue le premier gradin de cette architecture.

Symétriquement, l'obtention et l'assemblage des informations nécessaires font l'objet d'une entité fonctionnelle (gradin) séparée, car on rappelle que ce n'est pas le

30

rôle de la logique applicative. Le convertisseur XGate assure cette tâche en produisant un flux de données de résultat dans le langage XML. Ce langage a été naturellement choisi dans cet exemple comme le langage fédérateur : sa souplesse permet en effet de définir la structure de balisage la plus appropriée aux besoins de telle ou telle application.

Le convertisseur XGate permet d'ajouter un niveau de modularité essentiel : la collecte et la normalisation des données hétérogènes. En d'autres termes, la logique applicative ne fait que spécifier en XML ses besoins, via un script XF de conversion requête/résultat. Cette approche en 4 gradins (le quatrième gradin étant au niveau du fournisseur de données) permet un développement aisé, facilement modifiable et réutilisable, de cet exemple de commerce électronique.

4.2.2. Conversion de HTML vers WML

15

La figure 5 illustre l'accès à partir d'un téléphone portable aux horaires de vol d'avions au départs et arrivée des principaux aéroports français. La faisabilité de cette application a été expérimentée avec succès sur trois modèles de téléphones portables (Nokia 7110, Motorola TimePort P7389, Siemens S35I - marques déposées). L'appel d'un numéro privé à partir d'un tel téléphone portable permet d'obtenir les horaires des vols, les retards et annulations, affichés lisiblement, et mis à jour minute par minute. Le site testé n'a bien sûr subi aucune modification (d'ailleurs, aucun contact n'a été pris avec les responsables de ce site).

25 Comme dans l'exemple précédent, le premier gradin est l'interface de présentation : il s'agit dans ce cas du téléphone lui-même. Il communique avec le réseau GSM en mode CSD (plus communément appelé DATA, par opposition au mode SMS - « Short Message Service » en terminologie anglo-saxonne).

30 Sans entrer dans les détails sortant du cadre de la présente invention, le deuxième gradin (adaptation et transport, WSP/WTP, à savoir « Wireless Session Protocol

specification/Wireless Transaction Protocol specification en terminologie anglo-saxonne) permet au convertisseur XGate de « voir » le téléphone comme un dispositif IP émettant une requête HTTP et attendant une réponse HTTP en retour. Ce gradin garantit une indépendance vis-à-vis de la technologie GSM utilisée, fait
 5 particulièrement important étant donné l'évolution rapide des technologies concernées.

Le convertisseur XGate se trouve au troisième gradin. Il dialogue avec le site concerné, déchiffrant les informations, soumettant d'autres requêtes jusqu'à obtenir
 10 l'information que le téléphone appelant demande. Il traduit alors la réponse en générant les balises WML nécessaires à l'affichage en clair de cette réponse sur l'écran du téléphone.

Responsable du paramétrage de cette transformation, le script de conversion XF est peu volumineux ; typiquement, il n'excède pas une page de code.

15

4.2.3. Transformation en flux

La figure 6 des dessins synthétise le fonctionnement du convertisseur XGate en termes de flux et d'interface ; ainsi il ne s'agit pas sur cette figure de modules
 20 fonctionnels.

Une possibilité intéressante offerte par la présente invention est d'effectuer le travail de conversion en flux continu. Cette caractéristique est responsable de la rapidité de la réponse : ainsi, si la transformation le permet, les premières données en sortie
 25 peuvent être produites avant d'avoir lu les données en entrée dans leur totalité.

Les arbres des documents d'entrée et de sortie, définis en spécification DOM, sont au cœur de la conversion. Les flèches représentent la transformation nœud à nœud de ces arbres, transformation pilotée via le module « Transformer » interprétant le script
 30 de conversion XF.

L'interface « Normalizer » participe à la construction de l'arbre en entrée. Il est important de noter ici que la technique en flux continu n'impose pas une construction complète de l'arbre comme une condition nécessaire au démarrage de la transformation : ainsi, ce n'est qu'au moment où la transformation demande une
 5 branche qui n'est pas encore construite que l'interface « Normalizer » lira suffisamment de données en entrée pour construire cette branche.

L'interface « Finalizer » est quant à elle chargée de fournir le flux de sortie, en parcourant l'arbre DOM résultant. Cette tâche est permanente : si l'une des branches
 10 est incomplète, alors seulement l'interface « Finalizer » attendra que la transformation de cette branche soit terminée. Notons une autre caractéristique de l'interface « Finalizer », à savoir la possibilité qu'elle a d'épurer le flux de sortie. Par exemple, les téléphones portables ne reconnaissent pas les codifications HTML des caractères accentués, car la Description Technique de Document (DTD) du langage
 15 WML n'inclut pas la codification des caractères accentués. L'interface « Finalizer » possède donc avantageusement un composant de conversion des caractères accentués en caractères correspondants non accentués.

4.2.4. Agrégation et transformation

20

La figure 7 des dessins, correspondant à un exemple simplifié, montre quelques unes des étapes de la transformation réalisée par le script de conversion XF pour l'application de commerce électronique décrite plus haut. L'interface « Normalizer » a créé les arbres DOM de trois documents XML résultant de la recherche :
 25 QUERYDOC (recherche du produit dans la base de donnée), DIRDOC (recherche d'adresses dans la base LDAP), HTML (document contenant les images). Le script de conversion XF construit le document résultant, RESDOC, par une sélection des nœuds appropriés dans chacun des 3 arbres. Cet exemple permet de se rendre compte à la fois de la complexité de l'opération, et, en conséquence, de la puissance du
 30 langage de conversion dans lequel est écrit le script XF qui, par sa syntaxe et ses fonctionnalités en facilite la programmation.

Le modèle du document RESDOC (schéma XML, ou au format DTD pour « Document Technical Description » en terminologie anglo-saxonne) est spécifié dans le deuxième gradin « Logique Applicative » de la figure 4. Le rôle du convertisseur XGate est de décharger cette application de la recherche et de l'assemblage de ce document, opération qui serait extrêmement complexe si une « programmation classique » avait été mise en œuvre.

Le schéma de la figure 7 sous-entend que les trois arbres sont indépendants, en ce sens que le convertisseur XGate les construit en parallèle. Mais dans la pratique, la construction des trois arbres en entrée est interdépendante. Par exemple, la recherche d'un produit dans la base de données renseigne le nom du distributeur (CPNY), permettant d'interroger la base LDAP sur ses coordonnées géographiques. Ce principe de chaînage (redirection) utilise les fonctionnalités du module « Broker » (voir figure 3). Il est intégré dans le script de conversion XF via les variables de services.

4.3. Structure d'un script de conversion XF

Comme déjà indiqué, le script de conversion XF se situe au cœur de la présente invention. On va présenter ici quelques unes de ses caractéristiques fondamentales.

4.3.1. Structure générale du script XF : Gabarits

Un script de conversion XF se compose d'une liste de procédures, chacune d'entre elles étant applicable à des nœuds du document qui satisfont une condition bien définie, comme par exemple « être un nœud de type 'paragraphe' du corps du document ». La condition et sa procédure associée sont appelées gabarits (« template »).

30

On adopte dans la suite les désignations suivantes :

Template A : pour tout nœud satisfaisant la condition A, faire (procédure A).

Template B : pour tout nœud satisfaisant la condition B, faire (procédure B).

.....

- 5 Template Z : pour tout nœud satisfaisant la condition Z, faire (procédure Z).

10 Sur le plan de la syntaxe, un script de conversion XF est lui-même un document en langage balisé. Chaque gabarit y est représenté par une paire de balises ouvrante et fermante, signifiant respectivement le début d'un nouveau gabarit et la fin de ce même gabarit. La condition associée est l'attribut de correspondance « Match » de la balise ouvrante ; la procédure à exécuter est le contenu compris entre la balise ouvrante et la balise fermante de l'élément « template ».

- 15 Ainsi, la clause « pour tout nœud paragraphe du corps du document , faire (procédure P) » s'écrit de la façon illustrée sur la figure 8.

20 Cette syntaxe est inspirée de la syntaxe du langage XSL, mais la comparaison s'arrête là. D'abord, la syntaxe du critère « Match » est bien plus simple, comme on le verra plus loin. Ensuite, contrairement à l'élément « gabarit » du langage XSL (« xsl : template »), l'élément gabarit de XF ne contient aucunes sous-balises. Le contenu de l'élément gabarit est une procédure en langage ECMAScript appelée « procédure-gabarit ».

4.3.2. Programmation des procédures-gabarits

25

Les explications qui suivent sont faites en référence aux spécifications ECMAScript et DOM, auxquelles on se référera pour tous les détails nécessaires.

30 Chaque procédure-gabarit représente une méthode d'un objet de la classe de nœuds Node, classe standard définie par « org.w3c.dom.Node » dans la spécification DOM. Plus précisément, le sujet (« this ») de chaque procédure-gabarit est l'objet nœud de

l'arbre DOM qui a satisfait la condition d'exécution (Match). Toutes les méthodes définies par DOM pour la classe Node sont applicables à l'objet sujet (« this ») de la procédure-gabarit.

5 Par exemple, dans la procédure P suivante :

```
XF.log.writeln(this.getNodeName());
```

10 qui est appelée pour tout paragraphe du document, l'appel de la fonction DOM « getNodeName() » appliquée à l'objet « this » retourne le mot « PARA ».

La description des opérations en « liste de gabarits » est une approche particulièrement bien adaptée à la conversion d'arbre. Cependant, les traversées d'arbres et les récurrences qu'elle peut impliquer ne sont pas intuitives. En créant la
15 notion de procédure-gabarit dont l'objet sujet (« this ») se trouve être le nœud courant, la compréhension des effets induits est grandement facilitée.

L'approche proposée selon la présente invention possède comme atout majeur une programmation naturelle de la conversion, qui conduit à un code relativement simple,
20 lisible, qui ne nécessite pas d'apprentissage tout en étant particulièrement puissant.

Le Script de conversion XF se compose donc d'une liste de procédures-gabarits, chaque procédure étant décrite par la balise « template ». Pour que la conversion s'effectue, il faut maintenant procéder à l'exécution de ces procédures, et l'on va
25 maintenant décrire la façon dont ces procédures sont appelées.

4.3.3. Appel des procédures-gabarits

C'est par la méthode « applyTemplate(select) » que les procédures-gabarits sont
30 appelées. Cette méthode est une des méthodes de l'objet global XF, et s'écrit donc :

`XF.applyTemplate(select)`

La méthode « `applyTemplate` » donne l'ordre de rechercher et d'exécuter tout gabarit applicable aux nœuds de l'arbre rencontrés pendant la traversée spécifiée par la
 5 valeur de l'argument de sélection « `select` ». Cet argument de sélection est dite « équation de traversée ».

Comme le critère conditionnel d'exécution « `Match` », l'équation de traversée utilise une syntaxe simple, proche de celle prônée par le groupe de travail TEI (pour « Text
 10 Encoding Initiative » en terminologie anglo-saxonne). En effet ; le langage de conversion XF ne nécessite pas une sélection complexe des nœuds. La puissance de ce langage XF provient naturellement de son aptitude à être programmé. Une programmation classique (opérations conditionnelles de type « `if... else` », variables, boucles) permet d'exprimer aisément une solution algorithmique, et d'éviter que
 15 l'équation de traversée et/ou la condition d'exécution soient les seuls chargés de cette responsabilité. (On notera ici que, si l'on tentait d'utiliser la syntaxe standard du langage XSL, on démontrerait l'inadaptation des équations XSL à résoudre de réelles conditions de conversion : le degré de complexité des équations de correspondance et de sélection augmenterait exponentiellement pour chaque nouvelle condition,
 20 nécessitant des heures de travail sans certitude absolue de la validité de l'équation obtenue).

La méthode « `applyTemplate` » engendre une véritable réaction en chaîne. Ainsi, depuis la procédure-gabarit du nœud-racine du document, elle appelle toutes les
 25 procédures-gabarits selon l'équation de traversée. Les procédures-gabarits qui satisfont la condition d'exécution « `Match` » sont activées, et, à leur tour, peuvent lancer une méthode « `applyTemplate` » qui appelle toutes les procédures de gabarits, etc.

30 Ce mécanisme puissant étant parfois difficile à contrôler, on prévoit avantageusement selon l'invention des routines de débogage du script écrit,

spécialement étudiées pour fournir les moyens rapides de corriger une erreur de récursivité.

- 5 Pour que le dispositif de réaction en chaîne démarre, la procédure-gabarit du nœud-racine du document doit être appelée : c'est la seule procédure qui soit appelée automatiquement.

4.3.4. Administration des variables - cloîtres, environnement de service

- 10 Le langage de conversion XF fournit des fonctions qui permettent la communication de variables entre procédures-gabarits. Ces variables sont dites "cloîtrées", c'est à dire qu'une procédure-gabarit peut créer son propre jeu de variables (dans son « cloître »), variables qui seront héritées par l'ensemble des procédures-gabarits qu'elle appelle. Une variable créée par un cloître ne peut être transmise au cloître
15 parent.

- Les variables cloîtrées peuvent être de toute nature : nombres entiers, chaîne de caractères, tableaux, et même objets ECMAScript (autrement appelés « Eléments Dynamiques »), ce qui permet de d'étendre considérablement la puissance de ce
20 mécanisme.

Une méthode « applyTemplate » constitue en outre un moyen pour passer un nombre quelconque de variables par argument.

- 25 La durée de vie des variables cloîtrées est le temps d'exécution du script de conversion XF dans lequel elles sont définies. Une session, cependant, comporte généralement plusieurs requêtes et donc plusieurs conversions qui peuvent avoir certaines variables en commun. Le langage de conversion XF fournit cette possibilité par le biais de variables dites d'environnement de service.

Initialisées et réactualisées par le module « Broker » (voir plus haut), la valeur des variables d'environnement dépend essentiellement du service appelant, et notamment du type de requête arrivant). Dans le cas de la transformation en code WML par exemple, l'identité de l'appelant et le modèle de téléphone sont des variables
 5 enregistrées dans l'environnement de service.

4.3.5. Ecriture du document résultat

La production du résultat est bien entendu une fonction fondamentale du script de
 10 conversion. La méthode « XF.result() » fournit un objet qui permet d'accéder au document en sortie. Par exemple, en donnant accès au nœud racine de ce document, l'écriture du document en sortie revient à mettre en œuvre les méthodes définies par le modèle DOM pour ajouter des nœuds dans un arbre. Ces méthodes d'accès aléatoire en écriture coexistent avec la méthode « XF.result().write() » qui additionne
 15 simplement un morceau de langage balisé au flux de sortie. Il faut noter que dans tout les scripts expérimentaux réalisés dans le cadre du développement de la présente invention, même les plus complexes, les méthodes d'accès aléatoire ont été rarement nécessaires.

20 4.3.6. Autres fonctions du langage de conversion XF

Le langage XF peut fournir d'autres fonctions telles que :

- * recherche de texte par utilisation d'expression régulières
- * niveau de trace de débogage,
- 25 * instanciation d'objets Java,

étant noté ici que l'on a intérêt à limiter le nombre de méthodes pour garder à ce langage sa simplicité et son pouvoir fédérateur.

30 4.4. Exemple concret d'un script de conversion XF

Les extraits de code qui sont représentés sur les figures 9 et suivantes illustrent certains des aspects décrits ci-dessus. Ces exemples ont été extraits d'une application de conversion réelle, à savoir la recherche d'horaires de vols d'avions et leur affichage sur un téléphone WAP.

5

Dans le cas particulier de cette conversion, applicable à de nombreux autres cas, l'analyse préalable du plan du site concerné de la Toile nous a amené à énoncer les règles suivantes (on omettra ici le choix de l'aéroport pas souci de simplification) :

- 10 * la plupart des pages du site sont organisées en « cadres » (« frames » en terminologie anglo-saxonne). Un tel cadre délimite une portion rectangulaire constituant une sous-partie de l'écran d'affichage de la page HTML. Introduit par la deuxième génération des logiciels standard de navigation, ce mécanisme conçoit la page HTML comme une mosaïque de cadres, collection de cadres ou « ensemble de
- 15 cadres » (« frameset » en terminologie anglo-saxonne), chacun de ces cadres ayant une adresse HTTP qui lui est propre. Trouver l'information recherchée à travers des cadres impose un examen du contenu de chacun d'entre eux. Pour toute page dans laquelle une balise « FRAMESET » est rencontrée, la requête doit être redirigée en demandant l'accès à la page correspondant à chacun des cadres composant
- 20 l'ensemble de cadres. En général, ces cadres sont créés dynamiquement par les programmes de type « cgi-bin » du site exploré (dans le présent exemple, les horaires sont mis à jour toutes les 3 minutes). Rien n'empêche que la page en retour ne contienne à nouveau une balise FRAMESET : il faut alors réitérer le processus.
- * Ces redirections doivent s'enchaîner jusqu'à ce que l'on trouve (dans l'ordre) l'une
- 25 des deux pages suivantes :
- soit la page qui propose le choix : horaires de départ ou d'arrivée ;
 - soit la page du tableau des horaires (ce que l'on cherche).
- * Lorsqu'il est accédé à la page « choix des horaires », le système propose ce choix sur le téléphone, attend la réponse de l'utilisateur, puis reprend le processus en
- 30 orientant la recherche vers l'horaire désiré (départ ou arrivée)

* Lorsque la page des horaires est enfin trouvée, le système convertit le tableau au format HTML de la page dans un format approprié pour son affichage sur l'écran du téléphone.

- 5 La conversion indiquée au dernier point ci-dessus est donc assortie d'une logique de navigation préalable. Dans l'architecture du convertisseur XGate, le module « Broker » est responsable de ces redirections successives. Pour ce qui concerne le déroulement de la conversion, le module « Broker » n'est pas visible. Le script de conversion XF, appelé pour toutes les pages auxquelles il est accédé, y fait appel
10 implicitement, de façon transparente et simple.

4.4.1. Gabarit de base

- Etant ici rappelé qu'un script XF est une suite de gabarits, le premier gabarit de ce
15 script est représenté sur la figure 9 des dessins.

Ce gabarit est appelé pour le nœud « HTML » de l'arbre du document d'entrée. S'agissant d'une page de la Toile, ce nœud est le nœud-racine du document, faisant de ce gabarit un « gabarit de base », qui est le premier appelé dans la liste des
20 gabarits qui composent le script de conversion XF.

- Toutes les instructions spécifiques à l'outil de conversion commencent par "XF.", qui, en langage Java, signifie que l'on réfère une méthode de l'objet XF. Cet objet fait implicitement de l'environnement du script. Par exemple, dans le corps du
25 gabarit, l'instruction « XF.trace() » demande à l'objet XF de produire une trace de débogage dans les fichiers de contrôle du déroulement (fichier dit « log »).

- L'instruction « XF.applyTemplates(...) » demande au script d'appeler toutes les procédures-gabarits correspondant aux nœuds décrits par l'équation de traversée :
30 « origin().descendant(all,(FRAMESET|BODY)) ». Cette équation utilise une syntaxe du type « XPointers ». Elle signifie : partant du nœud courant – « origin() » –

parcourir tous les nœuds qui en descendent – « descendant(all,...) » – et dont les noms sont soit « FRAMESET », soit « BODY ».

- 5 En effet, comme énoncé plus haut, seuls les nœuds dont le nom est « FRAMESET » ou « BODY » doivent être pris en considération dans le cas particulier de cette conversion. Les nœuds « FRAMESET » arrêtent la conversion pour demander l'accès à la page qu'ils réfèrent ; les nœuds de corps « BODY » contiennent quant à eux des résultats à convertir et afficher.

10 4.4.2. Chaînage des cadres : gabarit FRAMESET

On va examiner tout d'abord le gabarit appelé pour les nœuds FRAMESET, tel qu'illustré sur la figure 10.

- 15 La première instruction de cette procédure-gabarit, à savoir « XF.setVar("framed", 1) », crée la variable cloîtrée "framed" en lui assignant la valeur 1. On en expliquera plus loin la raison.

- 20 Les descendants directs (enfants) du nœud dont le nom est « FRAMESET » sont des nœuds dont le nom est « FRAME » : ce sont ces nœuds qu'il faut examiner afin de trouver celui qui décrit la page à laquelle il faut accéder. Ceci est effectué de la façon suivante : le modèle DOM définit la méthode « getChildNodes() » qui fournit la liste de tous les enfants d'un nœud donné. On sait que le sujet du script d'un template est le nœud DOM pour lequel ce template a été appelé : c'est l'objet « this ». La boucle
25 de recherche de la page est donc une boucle « for » classique en soi, qui est désignée par la référence 101 sur la figure 10.

- Le modèle DOM définit que la liste obtenue est un objet (ici de la classe standard « org.w3c.dom.NodeList ») dont la méthode « item(i) » permet d'en extraire le 1^{ère}
30 élément. A l'intérieur de la boucle « for », l'instruction désignée en 102 sur la figure 10 permet d'obtenir successivement tous les enfants du nœud « FRAMESET » (à

savoir les nœuds « FRAME »), jusqu'à ce qu'il n'en reste plus (la variable « child » prend alors la valeur « null ») ou que le nœud « FRAME » en question soit celui que l'on recherchait.

- 5 Quant aux critères utilisés, l'examen du script de la figure 10 montre que seuls les cadres nommés : « HOME », « MainMenu », « Content », ou « Result » doivent être sujets à une opération de redirection. C'est cet ensemble de conditions qui, exprimées par les instructions « if (...) » du script de la figure 10, vont piloter l'opération de redirection.

10

On notera ici que les noms de ces cadres dépendent du choix du programme du site, qui peut évoluer avec le temps, mais l'on peut en général tabler sur le fait que les noms des cadres risquent moins d'être modifiés que d'autres éléments des pages. Le choix de critères basés sur les noms des cadres semble donc être le meilleur. En
15 outre, on pourra imposer à l'administrateur d'un site de ne pas effectuer de modifications qui pourraient compromettre le fonctionnement du script de conversion, sachant que l'obligation de figer un nom de cadre n'est pas pour lui une contrainte réelle. Ainsi une telle contrainte ne gêne pas les évolutions futures du site dans le temps, et les adaptations nécessaires du script de conversion restent aisées.

20

La redirection est déclenchée par l'instruction « XF.redirectTo(...) », méthode qui prend en paramètre l'adresse HTTP de la nouvelle page. Cette adresse est la valeur de l'attribut « src » du nœud « FRAME », obtenu par la méthode « getAttribute() » appliquée dans le modèle DOM au nœud « child » courant, à savoir le cadre.

25

La redirection vers une autre page relance le script de conversion avec les données (l'arbre en spécification DOM) de la nouvelle page. Par défaut, le même script de conversion est appelé. (On notera ici que le module « Broker » permet de gérer plusieurs scripts à la fois. Ainsi la méthode « XF.redirectTo() » comporte un
30 paramètre optionnel indiquant, si besoin est, le nom du script XF qui doit être exécuté. Ce nom est la valeur de l'attribut « name » de la balise-racine <xf:doc> qui



préside à tout script XF. En terminologie XF, un changement de script s'appelle « changement de mode ». Allié au chargement dynamique de fichiers de conversion « XF.load() », la possibilité de script multiples (gérés par le module « Broker » donne une souplesse remarquable dans l'administration des conversions.)

5

La navigation de nœud « FRAME » à nœud « FRAME » se poursuit jusqu'à ce qu'on accède aux pages pertinentes. Ces pages ne comportent pas de « FRAMESET » et leur balise « BODY » contient les informations que l'on recherche.

10

4.4.3. Conversion : gabarit BODY

Le gabarit « BODY » est illustré sur la figure 11 des dessins.

15 En premier lieu, les instructions indiquées en 111 et 112 sur la figure 11 écrivent dans le fichier de résultat respectivement le prologue et l'épilogue communs à tout document WML. Utilisées fréquemment, les variables « prolog » et « epilog » sont définies une fois pour toutes par le programmeur comme des chaînes de caractères (« string ») fixes.

20

Notons que l'exécution de ce gabarit est conditionnée par l'examen de la variable « framed » ; c'est l'instruction désignée par la référence 113 en figure 11.

25 Si cette condition est vraie, alors on sait que ce nœud « BODY » n'est pas dans une page ; en effet, s'il en avait été autrement, le gabarit « FRAMESET » aurait créé la variable « framed » (qui n'aurait donc pas la valeur null).

30 Le corps BODY d'une page formée de cadres est sans intérêt pour la présente conversion. Plus précisément, dans le langage HTML, une balise BODY peut suivre la balise FRAMESET, mais le contenu de ce BODY est dans ce cas sans intérêt, car il est utilisé par les logiciels de navigation datant d'avant l'époque où le mécanisme

de cadres a été introduit. Le plus souvent, une telle balise BODY affiche un texte indiquant l'incapacité du logiciel de navigation à traiter des documents comportant des cadres.

- 5 Dans ce cas, aucune autre instruction de ce gabarit n'est exécutée. Aucun autre gabarit n'étant actif, la conversion est terminée à ce stade.

Dans le cas contraire, cette page contient soit le menu de l'aéroport, soit l'un des horaires demandés. L'examen de l'adresse HTTP (URL) de la page permet de
10 l'indiquer : dans le présent exemple, si la page contient « HomePage », il s'agit du menu ; si elle contient « DayFlight », alors il s'agit d'un horaire.

L'adresse de la page courante fait partie des variables d'environnement : on l'obtient par la méthode XF « getURL() ». Cette méthode produit une chaîne de caractères –
15 et plus précisément un objet ECMAScript de la classe standard « String » – dont la méthode « indexOf () » permet de connaître s'il contient ou non la chaîne de caractères « DayFlight ».

Si c'est le cas, la page contient les horaires, sous forme de tableau HTML : il
20 convient d'examiner le corps de ce tableau, c'est à dire le nœud nommé TBODY (pour « table body »), quelque part dans la descendance du nœud BODY sujet de ce gabarit. C'est ce qu'indique l'équation de navigation désignée par 114 sur la figure 11.

25 4.4.4. Navigation hypertexte

La conversion du tableau des horaires sera décrite plus loin. En premier lieu, on examine le contenu de la clause « else » désignée par la référence 115 sur la figure 11, correspondant au cas où l'adresse de la page ne contient pas la chaîne de
30 caractères « DayFlight ».

Dans ce cas, il s'agit donc de la page d'accueil de l'aéroport. L'intervention de l'appelant au téléphone est alors nécessaire : veut-il les horaires de départ ? d'arrivée ? Il faut proposer ces deux choix, et, selon la réponse, naviguer vers la page adéquate. Sur un téléphone WAP, ce résultat est obtenu par l'envoi de la page en langage WML telle qu'illustrée sur la figure 12.

La procédure-gabarit appelante (gabarit BODY) a déjà écrit le prologue de cette page WML, et en écrira l'épilogue au retour. Il faut donc générer les deux lignes comportant la balise `<a>`, c'est-à-dire greffer sur l'arbre de sortie deux nœuds ayant des balises `<a>`.

Les balises `<a>` sont les points d'ancrage de la navigation hypertexte. Leur syntaxe WML est très proche de la syntaxe HTML, si ce n'est que la syntaxe WML impose la lettre « a » minuscule. Lorsque l'appelant activera le mot « Départs » affiché sur son téléphone WAP, il déclenchera une navigation vers la page qui se trouve à l'adresse HTTP « ddddd ».

Les valeurs des attributs href : "http://dddd" et "http://aaaaa" sont bien entendu données à titre illustratif. Dans la réalité, il s'agit des adresses HTTP des pages de la Toile qui vont permettre de poursuivre la navigation vers les horaires soit de départ, soit d'arrivée (respectivement). Ces valeurs apparaissent dans la page HTML en cours de conversion comme étant l'attribut source des balises `<A>` dans le code HTML de la page, qui sont donc des nœuds de l'arbre dans la descendance du nœud courant « BODY ».

On utilise la lettre « A » majuscule pour la balise HTML pour bien différencier la balise à rechercher dans la page HTML du document d'entrée de la balise WML `<a>` qui doit être générée dans l'arbre de sortie.

Il faut maintenant localiser les nœuds dont les balises sont `<A>`. L'examen du code de la page HTML conduit à remarquer que de tels nœuds ont comme enfant un nœud

de type « IMG » (une image) dont l'attribut « src » contient le mot « DEPART » ou le mot « ARRIVEE ». Il faut donc d'abord examiner tout les nœuds « IMG », c'est-à-dire appliquer les gabarits aux descendants de type « IMG », ce qui s'écrit comme indiqué à la deuxième ligne de l'instruction 115 de la figure 11.

5

Le gabarit associé à ces nœuds images est illustré sur la figure 13. On notera ici que, dans la pratique, l'équation de correspondance « match="IMG" » pourra être plus complexe en restreignant les nœuds candidats IMG à seulement ceux dont le père est un nœud de type A.

10

Par rapport à ce qui précède, il n'y pas réellement de notion nouvelle introduite par ce gabarit, si ce n'est l'appel à la fonction « addAnchor() ».

15 Cette fonction n'est pas la méthode d'un objet (sinon, sa syntaxe serait « sujet.addAnchor() »). Ce n'est pas non plus une instruction définie par ECMAScript. En effet, il s'agit d'une fonction dite « locale », définie par le programmeur du script XF en question.

20 On notera ici que de telles fonctions locales (ou routines) sont un outil précieux pour le programmeur à des fins de structuration, de réutilisation et de lisibilité du code. Les scripts XF offrent en eux mêmes cette possibilité. Ces routines, qui peuvent être appelées depuis n'importe quel gabarit, apparaissent dans le contenu de la balise : <xf:script>. Cette balise complète la liste des balises apparaissant dans le script XF.

25 On notera ici qu'il existe dans le présent exemple une autre balise, <xf:init>, qui apparaît dans un seul script XF, à savoir le script "par défaut". Le script XF par défaut est un script dont la balise-racine <xf:doc> ne possède pas d'attribut de nom "name", ou, s'il existe, possède un tel attribut dont la valeur est le nom réservé "#default". Le script par défaut est appelé lorsque le mode courant n'est pas défini.

30 La balise <xf:init> définit alors la procédure de conversion à appliquer lorsque le document source à convertir n'existe pas. C'est ce qui se produit lors du premier

contact avec un nouvel appelant au téléphone – d'où le nom « init ». C'est aussi ce qui peut se produire en cas d'erreur de navigation (code HTTP 404).

Un nombre quelconque de routines locales ou de variables locales peuvent être
5 définies. On y trouvera par exemple les variables locales chaînes de caractères prolog et epilog qui ont été évoquées plus haut.

Dans le cas d'espèce, l'élément « xf:scripts » contient la fonction « addAnchor() » et se présente de la façon illustrée sur la figure 14.

10

Le rôle assigné par le programmeur à cette routine est de construire un point d'ancrage : l'activation de ce point depuis le téléphone WAP entraînera la navigation vers la page désirée de la Toile, ici les horaires de départ ou les horaires d'arrivée.

15 Cette routine contient une boucle qui recherche le premier nœud « A » dans la lignée parentale du nœud « IMG » courant. Lorsque ce nœud a été trouvé, la routine appelle la méthode « XF.result().writeAnchor() ».

On va décrire le rôle de cette méthode en la décomposant.

20

Tout d'abord, la méthode « XF.result() » permet d'accéder au fichier résultat, à savoir le document qui sera envoyé vers le téléphone WAP. Il s'agit ici d'une écriture séquentielle, mais comme on l'a vu plus haut, il existe d'autres moyens pour écrire ce fichier résultat, tels qu'une écriture en accès aléatoire, à savoir l'ajout d'un
25 nœud de l'arbre DOM de sortie.

Ensuite, la méthode « writeAnchor() » de l'objet « fichier résultat » obtenu par la méthode « XF.result() » possède comme premier argument le texte du point d'ancrage associé à la référence hypertexte : « Départs », ou « Arrivées ». Le second
30 argument est la référence hypertexte, obtenue par la lecture de la valeur de l'attribut



« href » du nœud parent de balise <A>. Le troisième argument est la référence locale, qui est le nom qu'il faut associer à la référence hypertexte.

Des exemples de valeurs de ces trois arguments sont illustrés sur la figure 15.

5

pour obtenir le résultat, on écrit dans le fichier de sortie :

```
<a href="/xxx/airport.21">Départs</a>
```

- 10 « xxx » identifiant la session (l'appelant), « airport » identifiant le mode de transformation (<xf :doc name = "airport">, et « 21 » identifiant l'étape. Sans entrer dans les détails, c'est le rôle du module « Broker » de reconstituer l'adresse réelle de la page qui fournit le résultat et d'indiquer les autres références (appelant, mode, etc..) via les variables d'environnement du script de conversion XF appelé pour la
- 15 conversion de cette page.

4.4.5. Affichage du tableau des horaires

- On va maintenant décrire la conversion du résultat, à savoir le tableau des horaires
- 20 des vols qui correspond au nœud TBODY.

On indique tout d'abord une représentation de l'arbre du document à ce nœud. C'est là une description classique de la structure commune à tout tableau d'un document HTML. Cette représentation est illustrée sur la figure 16.

25

Les enfants du nœud TBODY sont les lignes de la table (nœud TR) . Chacune de ces lignes correspond ici à un vol. Les nœuds TR ont pour enfant des nœuds TD qui représentent les colonnes de la table. A leur tour, les nœuds TD ont pour descendants le texte même de chacune des colonnes. Ce texte donne les caractéristiques du vol.

30

On notera ici que le lien de descendance entre un nœud TD et le texte associé peut-être indirect, avec, par exemple, l'intervention d'un nœud indiquant la police de caractère utilisée.

- 5 Pour atteindre le texte de chacune des colonnes, on utilise donc l'équation de navigation :

```
origin().child(i).descendant(all,#text)
```

- 10 « i » étant une variable indiquant le numéro de la ligne. Cette équation signifie : partant du nœud courant TBODY – "origin()" – atteindre le ième nœud-enfant TR – "child(i)" – et appliquer le gabarit correspondant à tous les descendants de type text – "descendant(all,#text)".

- 15 Le gabarit TBODY utilise donc une boucle d'itération sur toutes les lignes de la table et s'écrit de la façon illustrée sur la figure 17.

- 20 Dans la réalité, il peut se produire que les lignes du tableau ne contiennent pas uniquement les renseignements afférents à chaque vol. Ainsi dans le présent exemple, certaines lignes sont utilisées pour la mise en page : elles sont repérées par un fond blanc, et sont suivies de 2 lignes de titre. C'est ce qui est répercuté dans le script par la condition désignée par la référence 171 sur la figure 17.

- 25 Les lignes qui nous intéressent font l'objet d'un appel à la procédure-gabarit chargée de la mise en forme du contenu des colonnes (les descendants de type texte). C'est le rôle de l'instruction désignée par la référence 172 sur la figure 17.

- 30 Le premier argument de la méthode « applyTemplates » est l'équation de navigation indiquée plus haut. Il existe un second argument « data ». Cet argument optionnel est passé tel quel à toutes les procédures-gabarits appliquées.

Pour cet argument « data », le programmeur de ce script a choisi un objet chargé de collecter les renseignements contenus dans chaque ligne du tableau. C'est un objet ECMAScript classique, créé classiquement par l'opérateur « new » et désigné par la référence 173 sur la figure 17.

5

La définition de l'objet « FlightData » est locale – il est créé par le programmeur du script XF – et se trouve donc dans le contenu de la balise : <xf :scripts>. S'agissant d'un dispositif standard d'ECMAScript, on n'entrera pas ici dans le détail de la construction de cet objet. Il suffit d'indiquer que l'objet FlightData fournit les méthodes pour collecter les informations associées à chaque ligne décrivant un vol :
 10 date, heure, aéroport, numéro du vol, compagnie, etc. En passant cet objet à la procédure gabarit de tout nœud de type texte de la ligne du tableau, chaque champ de l'objet « data » sera complété au fur et à mesure. Lorsque tous ces procédures auront été appliquées, à savoir que la méthode « applyTemplate() » est terminée,
 15 l'instruction désignée par la référence 174 sur la figure 17 produira sur le fichier de sortie le contenu textuel ainsi collecté.

20

La procédure-gabarit pour chaque portion de texte s'écrit simplement de la façon illustrée sur la figure 18.

4.4.6. Résultats

La figure 19 montre le résultat obtenu sur un téléphone WAP de marque Nokia, en parallèle avec les informations correspondantes identiques affichées sur un
 25 navigateur de Toile standard.

4.4.7. Trace de contrôle

La trace de contrôle (« log » en terminologie anglo-saxonne) est un fichier de
 30 contrôle qui permet de suivre les étapes de la transformation. La trace générée pour l'exemple qui vient d'être décrit comprend les éléments suivants :

- conversion 1 : enregistrement des messages de corrections de l'analyseur syntaxique ; le résultat de la conversion est une redirection (figure 20) ;
 - conversion 2, 3 et 4 : autres redirections (figures 21 à 23 respectivement) ;
 - conversion 5 : la page d'accueil de l'aéroport a été trouvée ; le script de conversion
- 5 XF renvoie une page au format WML vers le téléphone WAP ; cette page comporte deux liens hypertexte, respectivement vers les Départs et vers les Arrivées. On notera ici la construction « \$(sid) » qui permet d'identifier l'utilisateur entre chaque requête. La notation « \$(x) » est quant à elle typique des téléphones WAP : elle indique une
- 10 référence à la variable interne « x » que XF a généré dans l'électronique du téléphone. Le téléphone se connaît et s'identifie auprès du module « Broker » du convertisseur XGate (figure 24) ;
- conversion 6 : l'utilisateur du téléphone s'intéresse aux départs ; cette information nécessite une redirection (figure 25) ;
 - conversion 7 ; nouvelle redirection (figure 26) ;
- 15 - conversion 8 et fin : le tableau des horaires de départ a été converti au format WML, et est envoyé au téléphone WAP (figure 27).

4.5. Définition formelle du document XF : DTD

- 20 Comme on l'a vu plus haut, le script de conversion XF est encapsulé dans un document balisé XML. A ce type de document correspond donc correspondre une « Description Technique de Document » (DTD) qui est représentée sur la figure 28.
- 25 Bien entendu, la présente invention n'est nullement limitée aux formes de réalisation décrites ci-dessus et représentées sur les dessins, mais l'homme du métier saura y apporter de nombreuses variantes et modifications.

En particulier, l'homme du métier saura adapter les principes de l'invention aux

30 nouvelles spécifications susceptibles d'apparaître dans la définition des documents

échangés par un réseau tel que l'Internet, dans la programmation et dans la modélisation d'objets, etc.

REVENDICATIONS

1. Système de conversion de documents, destiné à transformer un premier document existant dans un premier format balisé à structure arborescente de nœuds
5 en un second document dans un second format balisé à structure arborescente de noeuds, caractérisé en ce qu'il comprend :
- un ensemble de procédures-gabarits (Template), chaque procédure-gabarit dudit ensemble contenant une première information de sélection d'une étendue de parcours de la structure arborescente du premier document, une seconde information
10 de sélection d'un type de balise du premier document auquel la procédure gabarit peut s'appliquer, et un ensemble d'une ou plusieurs actions, au moins certaines procédures gabarits contenant des actions de collecte d'informations délimitées par les balises dont le type correspond à celui défini par la seconde information de sélection de ces procédures gabarits,
- 15 un programme contenant des instructions standard (applyTemplates) pour sélectivement appeler lesdites procédures gabarits,
- des moyens de traitement répondant audit programme et aptes, lors de l'appel d'une procédure-gabarit, à parcourir les nœuds de la structure arborescente inclus dans son étendue telle que définie par la première information de sélection de
20 ladite procédure-gabarit, et pour chaque nœud parcouru, à déterminer si le type de balise correspondant audit nœud correspond à la seconde information de sélection et, dans l'affirmative, à exercer la ou les actions correspondantes, et
- des moyens aptes, à partir des actions de collecte contenues dans les procédures-gabarits exécutées, pour construire le second document.
- 25
2. Système selon la revendication 1, caractérisé en ce que les actions d'au moins certaines procédures-gabarits contiennent des instructions d'appel d'autres procédures gabarits.

3. Système selon la revendication 2, caractérisé en ce que la première information de sélection d'étendue d'une procédure-gabarit est basée sur un nœud de départ constitué par le nœud à partir duquel ladite procédure-gabarit est appelée.
- 5 4. Système selon l'une des revendications 2 et 3, caractérisé en ce qu'au moins certaines procédures-gabarits contiennent des actions de création de variables temporaires cloîtrées, et en ce que chacune de ces variables cloîtrées est héritée par toute procédure-gabarit appelée par de telles procédures-gabarits.
- 10 5. Système selon l'une des revendications 1 à 3, caractérisé en ce que les procédures-gabarits comprennent une procédure-gabarit de base possédant la seconde information de sélection correspondant à un nœud racine de la structure arborescente du premier document, et en ce que les actions de ladite procédure-gabarit de base comprennent un appel de procédures-gabarits.
- 15 6. Système selon la revendication 5, caractérisé en ce que les actions de ladite procédure-gabarit de base comprennent un appel de procédures-gabarits dont la seconde information de sélection correspond à un type de balise « corps ».
- 20 7. Système selon la revendication 6, caractérisé en ce que la structure arborescente de nœuds du premier document comprend des nœuds de type « cadre », et en ce que les actions de ladite procédure-gabarit de base comprennent un appel de procédures-gabarits dont la seconde information de sélection correspond à un type de balise « ensemble de cadres ».
- 25 8. Système selon l'une des revendications 1 à 7, caractérisé en ce qu'au moins certaines procédures-gabarits comprennent au moins une action de redirection vers un premier document différent, sur lequel lesdits moyens de traitement sont appliqués sur la base du même programme.

9. Système selon l'une des revendications 1 à 8, caractérisé en ce qu'au moins certaines procédures gabarits comprennent au moins une action conditionnelle basée sur le contenu d'une adresse d'accès au premier document.
- 5 10. Système selon l'une des revendications 1 à 9, caractérisé en ce qu'au moins certaines procédures-gabarits comprennent au moins une action constituant une méthode d'un objet et/ou au moins une action constituant une fonction locale programmée.
- 10 11. Système selon la revendications 1 à 10, caractérisé en ce que certaines procédures-gabarits comprennent une fonction locale d'ancrage apte à convertir une requête adaptée à la structure du second document en une adresse d'un premier document contenant l'information recherchée.
- 15 12. Système selon l'une des revendications 1 à 11, caractérisé en ce que les moyens de construction du second document comprennent des actions d'écriture contenues dans certaines procédures-gabarits dont la seconde information de sélection définit une balise à contenu, lesdites actions d'écriture étant aptes à assembler de façon prédéterminée au moins une partie des contenus desdites balises.
- 20 13. Système selon l'une des revendications 1 à 12, caractérisé en ce que les premiers documents comprennent des pages structurées dans un langage balisé standard adapté à une consultation sur poste informatique client via l'Internet et en ce que les seconds documents comprennent des pages structurées dans un second
- 25 langage balisé standard adapté à une consultation sur appareil miniature portable.
14. Système selon l'une des revendications 1 à 13, caractérisé en ce que les moyens de traitement et de construction du second document opèrent en mode dynamique au cours d'une session entre un appareil apte à afficher des informations
- 30 dans la structure des seconds documents et un serveur apte à fournir des informations dans la structure des premiers documents.

15. Système selon l'une des revendications 1 à 14, caractérisé en ce qu'il constitue un gradin d'une architecture multi-gradins.
- 5 16. Architecture informatique multi-gradins, caractérisée en ce qu'elle comprend en succession, reliés par réseau informatique, un premier gradin de consultation de données sur poste-client, un second gradin d'application sur serveur, un troisième gradin d'aggrégation de données comprenant un système de conversion selon l'une des revendications 1 à 14 et un quatrième gradin comprenant une pluralité de types de sources de données indépendantes.
- 10
17. Architecture selon la revendication 16, caractérisée en ce que la pluralité de types de sources de données indépendantes comprennent au moins deux types de sources parmi les serveurs Internet, les serveurs d'annuaires à accès rapide et les serveurs de bases de données à accès standard par requêtes.
- 15
18. Architecture informatique/téléphonique multigradins, caractérisée en ce qu'elle comprend en succession, reliés par réseau informatique et par réseau téléphonique sans fil, un premier gradin de consultation de données sur un appareil portable tel qu'un téléphone portable, un second gradin de transport de données sans fil, un troisième gradin comprenant un système de conversion de documents selon l'une des revendications 1 à 14 et un quatrième gradin comprenant au moins un type de source de données.
- 20
19. Architecture selon la revendication 18, caractérisé en ce que ledit type de source de données consiste en des pages structurées dans un langage balisé standard adapté à une consultation sur poste informatique client, lesdites pages constituant des premiers documents, et en ce que le système de conversion compris dans le troisième gradin est apte à construire des seconds documents dans un second langage balisé standard adapté à une consultation sur appareil miniature portable.
- 25
- 30

ORIGINAL

CABINET REGIMBEAU
CONSEILS EN BREVETS
26, Avenue Kléber
75116 PARIS

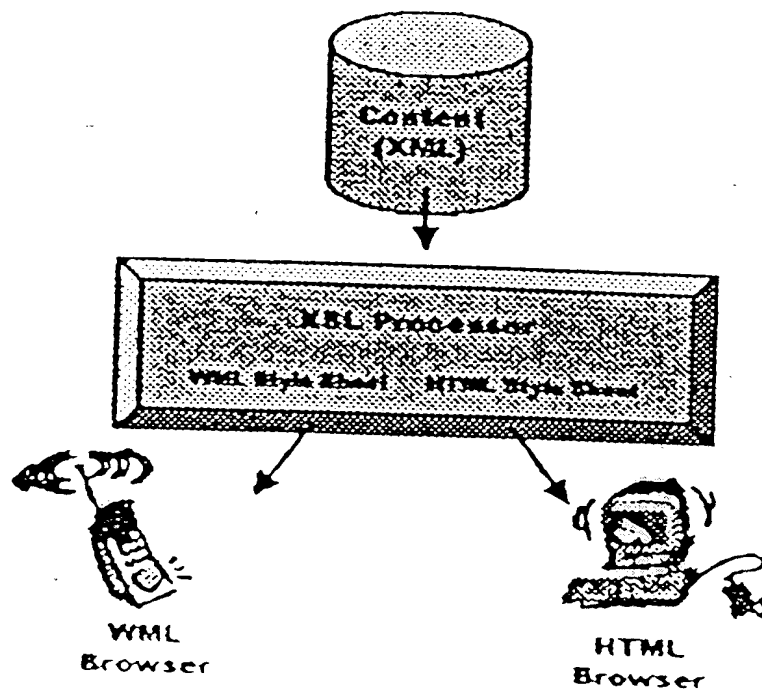
FIGURE 1

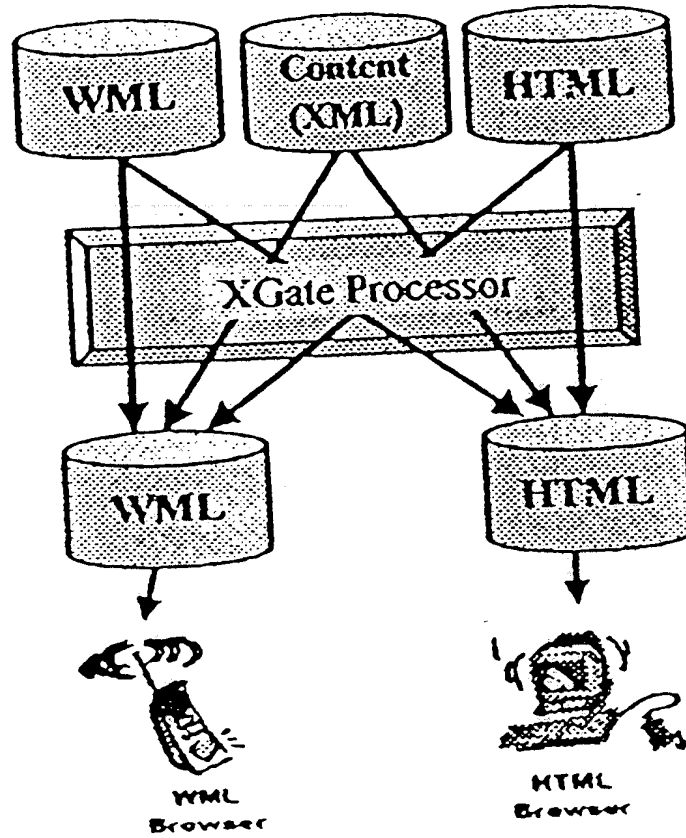
FIGURE 2

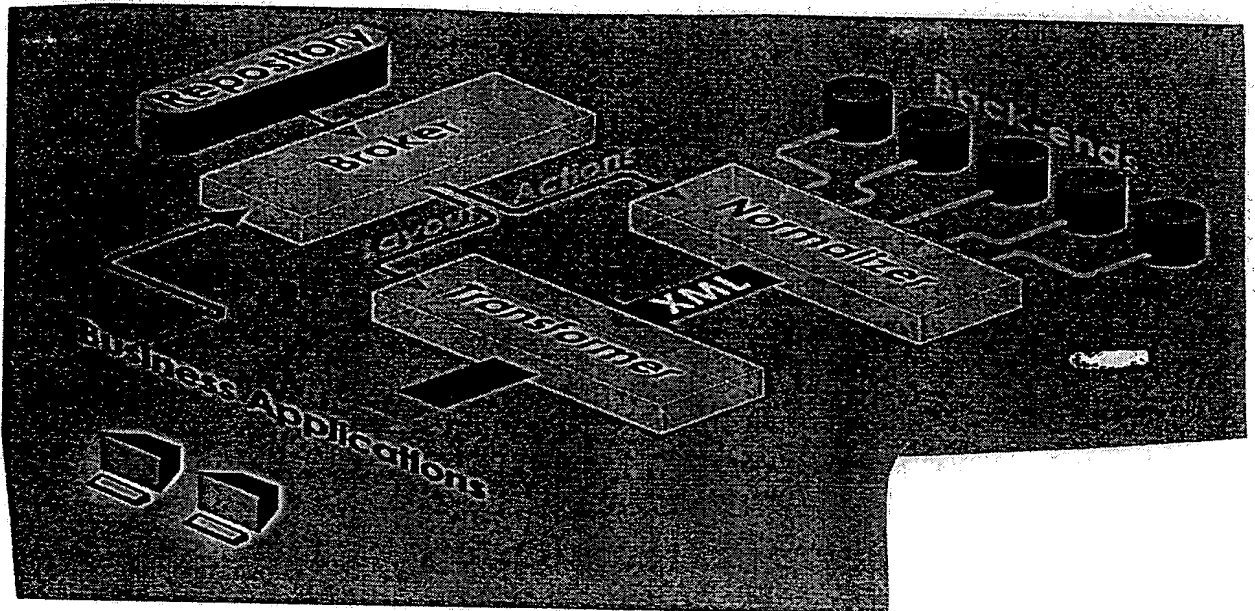
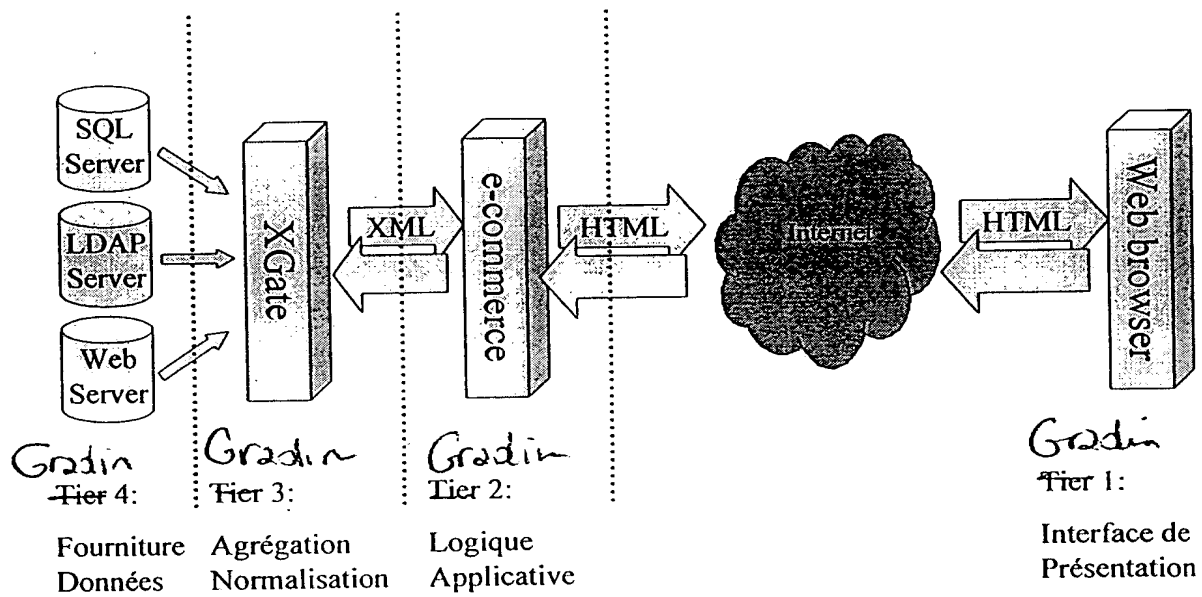
FIGURE 3

FIGURE 4

Application e-commerce



Conversion HTML WML

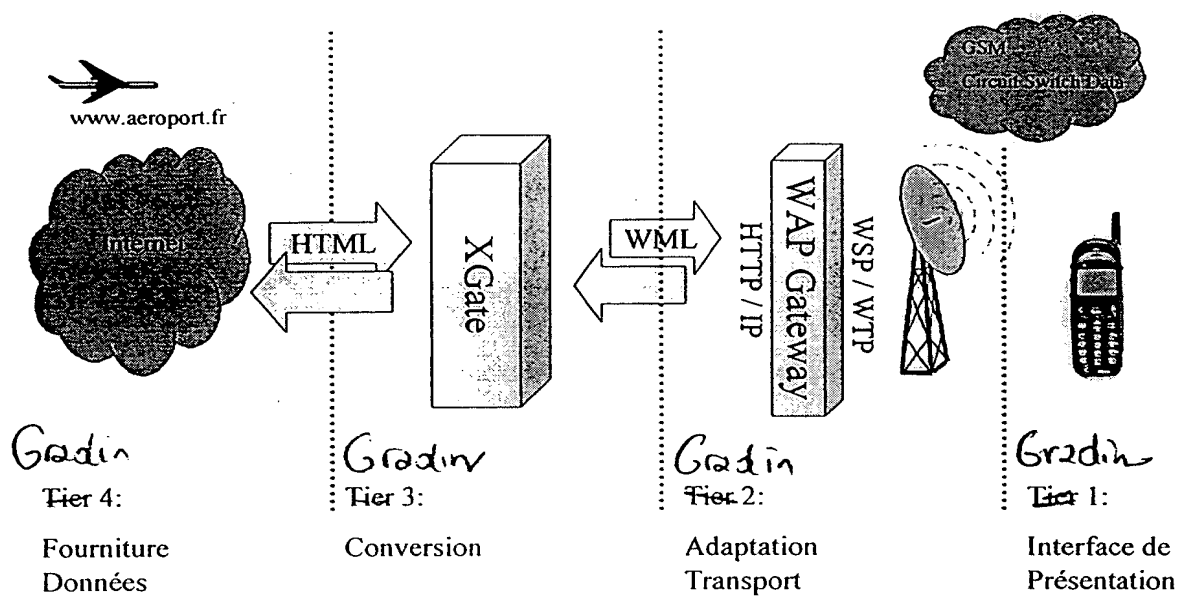


FIGURE 6

Transformation : flux

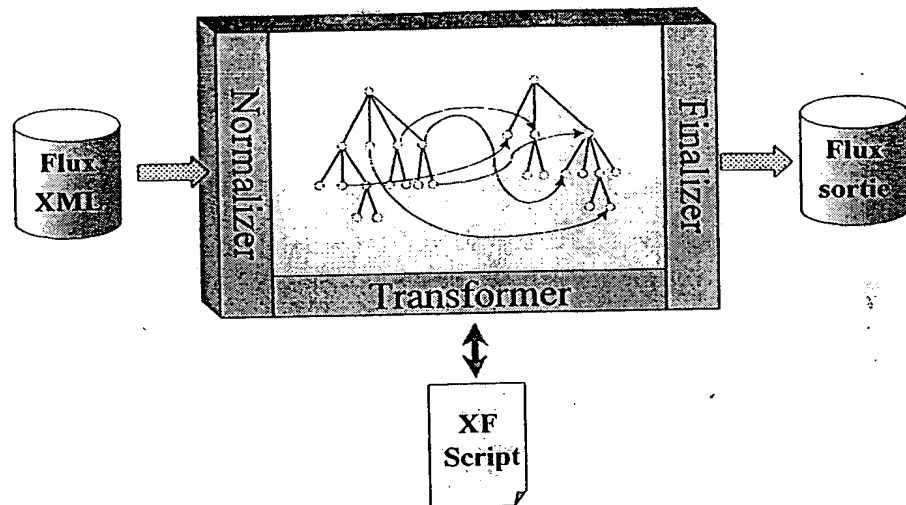


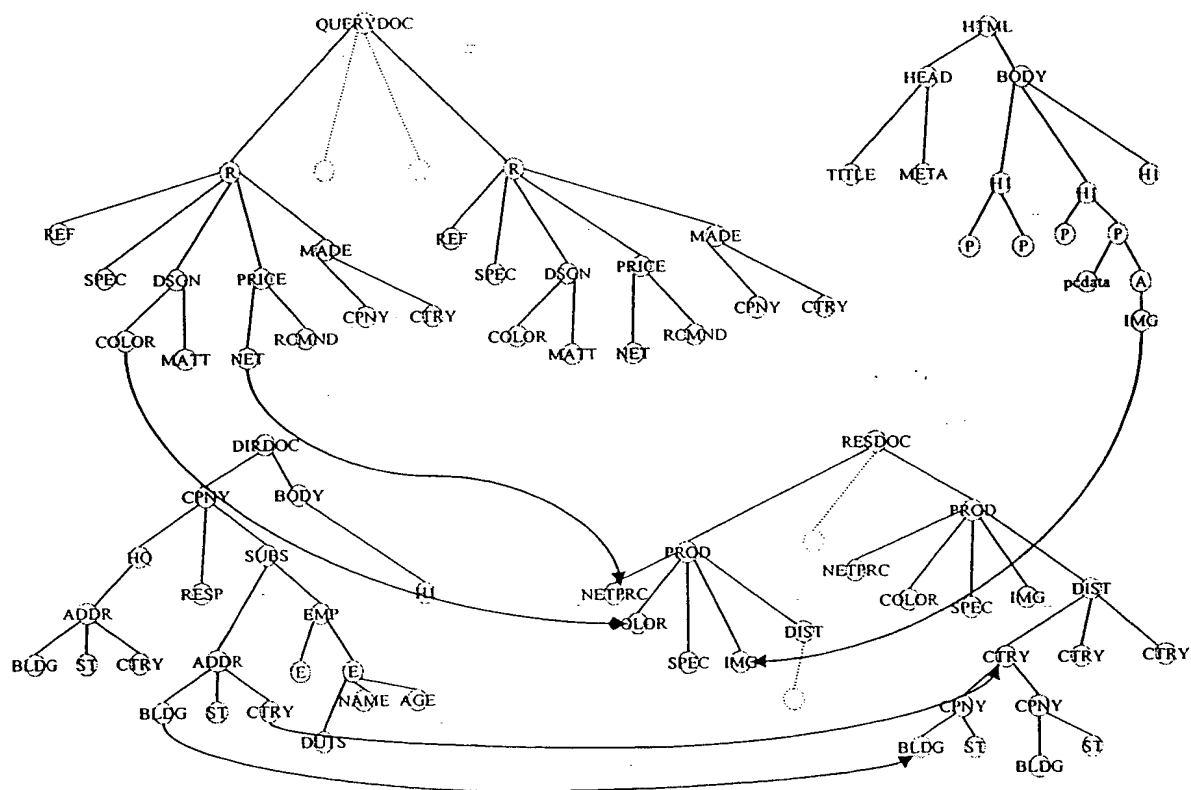
FIGURE 7**Agrégation et Transformation**

FIGURE 8

```
<xf:template match=□P□>  
(procédure P)  
</xf:template>
```

FIGURE 9

```
<xf:template match="HTML">  
  XF.trace();  
  XF.applyTemplates("origin().descendant(all, (FRAMESET|BODY))");  
</xf:template>
```

FIGURE 10

```
<xf:template match="FRAMESET">
  XF.setVar("framed", 1);
  var list = this.getChildNodes();
  for (var i=0; ;++i) {
    var child = list.item(i);
    if (child == null) break;
    if (child.getNodeName() == "FRAME") {
      var name = child.getAttribute("name");
      if (
        ((name == null) && (child.getNextSibling() == null)) ||
        (name == "HOME") || (name == "MainMenu") ||
        (name == "Content") || (name == "Result")
      ) {
        XF.redirectTo(child.getAttribute("src"));
        break;
      }
    }
  }
}</xf:template>
```

101

102

FIGURE 11

```
<xf:template name="body" match="BODY"> 113
  if (null == XF.getVar("framed")) {
    XF.result().write(prolog); 111
    var url = XF.getURL();
    if (url.indexOf("DayFlight") > 0) { // result
      XF.applyTemplates("origin().descendant(all, TBODY)"); 114
    } else { // HomePage (Dep/Arr)
      XF.applyTemplates("origin().descendant(all, IMG)"); 115
    }
    XF.result().write(epilog); 112
  }
</xf:template>
```

FIGURE 12

<pre><?xml version="1.0"?> <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml"> <wml> <card> <p> Départs Arrivées </p> </card> </wml></pre>	<p>prologue</p> <p>epilogue</p>
---	---------------------------------

FIGURE 13

```
<xf:template name="menuItem" match="IMG43">
  var text = this.getAttribute("src");
  if (text.indexOf("DEPART") > 0) {
    addAnchor(this, "Départs", "airport.21");
  } else if (text.indexOf("ARRIVE") > 0) {
    addAnchor(this, "Arrivées", "airport.22");
  }
</xf:template>
```

FIGURE 14

```
<xf:scripts>
function addAnchor(node, text, localRef) {
  while (null != (node = node.getParentNode())) {
    if (node.getNodeName() == 'A') {
      XF.result().writeAnchor(text, node.getAttribute("href"), localRef)
      break;
    }
  }
}
</xf:scripts>
```

FIGURE 15

Texte	Départs
Réf hypertexte	http://www.montpellier.aeroport.fr/cgi-bin/WebObjects.dll/Uccega/MPL?action=goDayDepartures
Réf locale	airport.21

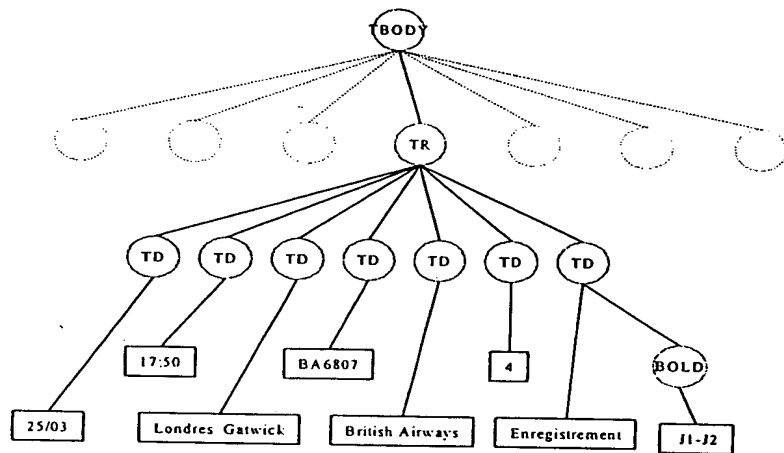
FIGURE 16**Affichage du tableau des horaires**

FIGURE 17

171

```
<xf:template match="TBODY">
  var list = this.getChildNodes();
  for (var i=0; ;++i) {
    child = list.item(i);
    if (child == null) break;
    if (child.getAttribute("bgcolor") == "FFFFFF") { // irrelevant
      i += 3;
    } else {
      var data = new FlightData();
      XF.applyTemplates(
        "origin().child(" + i + ").descendant(all,#text)",
        data
      );
      XF.result().write(data.timeFrom + "<br/>" + data.others);
    }
  }
}</xf:template>
```

173

172

Fig. 17

174

FIGURE 18

La procédure template pour chaque portion de texte s'écrit simplement:

```
<xf:template name="flight" match="text()">
  var text = this.getNodeValue().reduce();
  if (text.length > 0) {
    var i = arg0.item++;
    if (i == 1) { arg0.timeFrom = text + " "; // arrival time
    else if (i == 2) arg0.timeFrom += text; // from
    else if (i == 3) arg0.others = "." + text + "<br/>"; // airline
    else if (i == 4) arg0.others += "." + text + "<br/>"; // buildg
    else if (i == 5) arg0.others += "." + text + "<br/>"; // remarks
    else if (i == 6) arg0.others += "." + text + "<br/>"; // departure time
  }
</xf:template>
```

FIGURE 19

Nokia 7110

Vols
10:50 Paris-Cdg
AF7725
Enregistrement D1-E2
11:15 Lyon
Options Back

Microsoft Internet Explorer
http://scripte/WebObjects/Uccoga/NTE?action=goDepartures
Favoris Tools Help
Stop Refresh Home Search Favorites History Mail Print Edit Capture
www.aeroport.fr/scripts/WebObjects/Uccoga/NTE?action=goDepartures
Web Channel Guide Customized Links Free Mail Internet Explorer News Internet Start Windows
ROPORTS FRANCAIS Départs du jour
Samedi 17 Juin 11:39
Recherche (horaire, provenance, compagnie) Autre recherche

DESTINATION	N° VOL	COMPAGNIE	SALLE DEPART	OBSERVATIONS	HEURE ARRIVEE
Paris-Cdg	AF7725	Air France	2	Enregistrement D1-E2	
Lyon	AF7862	Air France	3B	Décollé 11:23	
Nice	SN4882	Sabena	1		
Nice	FU172	Air Littoral	1	Enregistrement F1-F2	
Marseille	AF5763	Air France	1	Enregistrement H1-H2	
Londres Gatwick	AF5852	Air France	4	Enregistrement I1-I2	
Bruxelles	SN3640	Sabena	1	Enregistrement G1-G2	
Bruxelles	VM1003	Regional Airlines	1		
Palma	2R478	Star Airlines	5		
Paris-Cdg	AF7727	Air France	3A		

DESTINATION	N° VOL	COMPAGNIE	SALLE DEPART	OBSERVATIONS	HEURE ARRIVEE
Paris-Cdg	AF7729	Air France	2		
Brest	AF5853	Air France	2		
Malaga	JK4534	Spanair, Sa	5		
Athenes	RN1602	Euralair International	3A		
Athenes	EY832	Aerolyon	5		
Marseille	FU765	Air Littoral	1		
Marseille	AF5765	Air France	1		
Londres Gatwick	AF5854	Air France	4		
Londres Gatwick	BA5807	British Airways	4		
Nice	FU176	Air Littoral	1		

17.06 17:00 Athenes EY832 Aerolyon 5
17.06 17:40 Marseille FU765 Air Littoral 1
17.06 18:35 Marseille AF5765 Air France 1
17.06 18:40 Londres Gatwick AF5854 Air France 4
17.06 18:50 Londres Gatwick BA5807 British Airways 4
17.06 19:20 Nice FU176 Air Littoral 1

Les vols suivants

Les horaires en temps réel des aéroports français déploie ses ailes

SAAI pour louer votre voiture en France rent a car anywhere in France

Ces horaires ont été mis à jour le 17/06/2000 à 11h30. Pour Internet

FIGURE 20

```

[Jun 17 12:38:55 CEST 2000] Service-47 started
[Jun 17 12:38:55 CEST 2000] Service-47, Proxy: plain copy
[Jun 17 12:38:55 CEST 2000] Service-47-in: GET http://www.Nantes.aeroport.fr HTTP/1.0
accept: text/html
proxy-connection: Keep-Alive
accept-charset: ISO-8859-1, UTF-8
host: www.Nantes.aeroport.fr
user-agent: JaxoXform (1.25)

[Jun 17 12:38:56 CEST 2000] Service-47-out: HTTP/1.0 200 OK
content-type: text/html
date: Sat, 17 Jun 2000 09:42:47 GMT
last-modified: Fri, 12 May 2000 14:53:47 GMT
accept-ranges: bytes
link: <http://www.aeroport.fr/?PageServices>; rel="PageServices"
content-length: 5932
server: Netscape-Enterprise/3.5.1

[Jun 17 12:38:56 CEST 2000] Service-47-out, Xform: parsing...
Cannot start element "META": a required element is missing in BODY
line 3, col 76
in:  '"Content-Type" CONTENT="text/html; charset=iso-8859-1">'
at: >-----*
Cannot start element "META": a required element is missing in BODY
line 4, col 451
in:  're, terminal, piste, runway, fret aerien, air freight">'
at: >-----*
Cannot start element "META": a required element is missing in BODY
line 5, col 403
in:  'mercial services, tourism in the Aquitaine region ...">  </head>'
at: >-----*
Warning: invalid end tag: "HEAD"
line 5, col 413
in:  'rvices, tourism in the Aquitaine region ...">  </head>'
at: >-----*
Warning: invalid end tag: "NOFRAMES"
line 60, col 12
in:  '</noframes>'
at: >-----*

[Jun 17 12:38:57 CEST 2000] Service-47-out, Xform: transforming...
[Jun 17 12:38:57 CEST 2000] Service-47-out, Xform: done.
[Jun 17 12:38:57 CEST 2000] Service-47 ended
[Jun 17 12:38:57 CEST 2000] Captured response -----
HTTP/1.0 302 Redirected by transform
location: http://xgate.jaxo.com/gjm/airport.rd
content-type: text/vnd.wap.wml; charset=ISO-8859-1
date: Sat, 17 Jun 2000 09:42:47 GMT
last-modified: Fri, 12 May 2000 14:53:47 GMT
accept-ranges: bytes
link: <http://www.aeroport.fr/?PageServices>; rel="PageServices"
server: Netscape-Enterprise/3.5.1

```

FIGURE 21

```
[Jun 17 12:38:58 CEST 2000] Service-48 started
[Jun 17 12:38:58 CEST 2000] Service-48, Proxy: plain copy
[Jun 17 12:38:58 CEST 2000] Service-48-in: GET
http://www.nantes.aeroport.fr/scripts/WebObjects/Uccega/NTE HTTP/1.0
accept: text/html
proxy-connection: Keep-Alive
accept-charset: ISO-8859-1, UTF-8
host: www.nantes.aeroport.fr
user-agent: JaxoXform (1.25)

[Jun 17 12:38:59 CEST 2000] Service-48-out: HTTP/1.0 200 NeXT
content-type: text/html
date: Sat, 17 Jun 2000 09:42:49 GMT
content-length: 321
server: Netscape-Enterprise/3.5.1

[Jun 17 12:38:59 CEST 2000] Service-48-out, Xform: parsing...
[Jun 17 12:38:59 CEST 2000] Service-48-out, Xform: transforming...
[Jun 17 12:38:59 CEST 2000] Service-48-out, Xform: done.
[Jun 17 12:38:59 CEST 2000] Service-48 ended
[Jun 17 12:38:59 CEST 2000] Captured response -----
HTTP/1.0 302 Redirected by transform
location: http://xgate.jaxo.com/gjm/airport.rd
content-type: text/vnd.wap.wml; charset=ISO-8859-1
date: Sat, 17 Jun 2000 09:42:49 GMT
server: Netscape-Enterprise/3.5.1
```

FIGURE 22

```
[Jun 17 12:38:59 CEST 2000] Service-49 started
[Jun 17 12:39:00 CEST 2000] Service-49, Proxy: plain copy
[Jun 17 12:39:00 CEST 2000] Service-49-in: GET
http://www.nantes.aeroport.fr/scripts/WebObjects/Uccega/NTE.woa/2093200000268020000023311
00000536431/Main.wo/7192100000436431/1/1/Mermoz HTTP/1.0
accept: text/html
proxy-connection: Keep-Alive
accept-charset: ISO-8859-1, UTF-8
host: www.nantes.aeroport.fr
user-agent: JaxoXform (1.25)

[Jun 17 12:39:05 CEST 2000] Service-49-out: HTTP/1.0 200 NeXT
content-type: text/html
date: Sat, 17 Jun 2000 09:42:52 GMT
content-length: 490
server: Netscape-Enterprise/3.5.1

[Jun 17 12:39:05 CEST 2000] Service-49-out, Xform: parsing...
[Jun 17 12:39:05 CEST 2000] Service-49-out, Xform: transforming...
[Jun 17 12:39:05 CEST 2000] Service-49-out, Xform: done.
[Jun 17 12:39:05 CEST 2000] Service-49 ended
[Jun 17 12:39:05 CEST 2000] Captured response -----
HTTP/1.0 302 Redirected by transform
location: http://xgate.jaxo.com/gjm/airport.rd
content-type: text/vnd.wap.wml; charset=ISO-8859-1
date: Sat, 17 Jun 2000 09:42:52 GMT
server: Netscape-Enterprise/3.5.1
```

FIGURE 23

```

[Jun 17 12:39:05 CEST 2000] Service-50 started
[Jun 17 12:39:06 CEST 2000] Service-50, Proxy: plain copy
[Jun 17 12:39:06 CEST 2000] Service-50-in: GET
http://www.aeroport.fr/Uccega/NTE/Html/FR/HP/home.htm HTTP/1.0
accept: text/html
proxy-connection: Keep-Alive
accept-charset: ISO-8859-1, UTF-8
host: www.aeroport.fr
user-agent: JaxoXform (1.25)

[Jun 17 12:39:10 CEST 2000] Service-50-out: HTTP/1.0 200 OK
content-type: text/html
date: Sat, 17 Jun 2000 09:42:57 GMT
last-modified: Wed, 17 May 2000 16:10:48 GMT
accept-ranges: bytes
link: <http://www.aeroport.fr/Uccega/NTE/Html/FR/HP/home.htm?PageServices>;
rel="PageServices"
content-length: 507
server: Netscape-Enterprise/3.5.1

[Jun 17 12:39:10 CEST 2000] Service-50-out, Xform: parsing...
Cannot start element "BODY": not permitted by the document structure.
  line 11, col 35
  in:  '<noframes><body bgcolor="#FFFFFF">'
  at: >-----*
[Jun 17 12:39:11 CEST 2000] Service-50-out, Xform: transforming...
[Jun 17 12:39:11 CEST 2000] Service-50-out, Xform: done.
[Jun 17 12:39:11 CEST 2000] Service-50 ended
[Jun 17 12:39:11 CEST 2000] Captured response -----
HTTP/1.0 302 Redirected by transform
location: http://xgate.jaxo.com/gjm/airport.rd
content-type: text/vnd.wap.wml; charset=ISO-8859-1
date: Sat, 17 Jun 2000 09:42:57 GMT
last-modified: Wed, 17 May 2000 16:10:48 GMT
accept-ranges: bytes
link: <http://www.aeroport.fr/Uccega/NTE/Html/FR/HP/home.htm?PageServices>;
rel="PageServices"
server: Netscape-Enterprise/3.5.1

```


FIGURE 24

```

[Jun 17 12:39:11 CEST 2000] Service-51 started
[Jun 17 12:39:12 CEST 2000] Service-51, Proxy: plain copy
[Jun 17 12:39:12 CEST 2000] Service-51-in: GET
http://www.aeroport.fr/Uccega/NTE/Html/FR/HP/2.htm HTTP/1.0
accept: text/html
proxy-connection: Keep-Alive
accept-charset: ISO-8859-1, UTF-8
host: www.aeroport.fr
user-agent: JaxoXform (1.25)

[Jun 17 12:39:12 CEST 2000] Service-51-out: HTTP/1.0 200 OK
content-type: text/html
date: Sat, 17 Jun 2000 09:43:03 GMT
last-modified: Wed, 17 May 2000 16:10:48 GMT
accept-ranges: bytes
link: <http://www.aeroport.fr/Uccega/NTE/Html/FR/HP/2.htm?PageServices>;
rel="PageServices"
content-length: 2603
server: Netscape-Enterprise/3.5.1

[Jun 17 12:39:13 CEST 2000] Service-51-out, Xform: parsing...
Cannot start element "BODY": a required element is missing in TITLE
  line 10, col 23
  in:   '<BODY bgColor=#ffffff>'
  at: >-----*

[Jun 17 12:39:13 CEST 2000] Service-51-out, Xform: transforming...
[Jun 17 12:39:13 CEST 2000] Service-51-out, Xform: done.
[Jun 17 12:39:13 CEST 2000] Service-51 ended
[Jun 17 12:39:13 CEST 2000] Captured response -----
HTTP/1.0 200 OK
content-type: text/vnd.wap.wml; charset=ISO-8859-1
date: Sat, 17 Jun 2000 09:43:03 GMT
last-modified: Wed, 17 May 2000 16:10:48 GMT
accept-ranges: bytes
link: <http://www.aeroport.fr/Uccega/NTE/Html/FR/HP/2.htm?PageServices>;
rel="PageServices"
server: Netscape-Enterprise/3.5.1
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml><template><do type="prev" label="Back"><prev/></do></template>
<card title="Montpellier"><p>
<a href="/$(sid)/airport.21">Départs</a>
<a href="/$(sid)/airport.22">Arrivées</a></p></card></wml>

```

FIGURE 25

```
[Jun 17 12:39:18 CEST 2000] Service-52 started
[Jun 17 12:39:18 CEST 2000] Service-52, Proxy: plain copy
[Jun 17 12:39:18 CEST 2000] Service-52-in: GET
http://www.aeroport.fr/scripts/WebObjects/Uccega/NTE?action=goDayDepartures HTTP/1.0
accept: text/html
proxy-connection: Keep-Alive
accept-charset: ISO-8859-1, UTF-8
host: www.aeroport.fr
user-agent: JaxoXform (1.25)

[Jun 17 12:39:26 CEST 2000] Service-52-out: HTTP/1.0 200 NeXT
content-type: text/html
date: Sat, 17 Jun 2000 09:43:17 GMT
content-length: 563
server: Netscape-Enterprise/3.5.1

[Jun 17 12:39:27 CEST 2000] Service-52-out, Xform: parsing...
[Jun 17 12:39:31 CEST 2000] Service-52-out, Xform: transforming...
[Jun 17 12:39:31 CEST 2000] Service-52-out, Xform: done.
[Jun 17 12:39:31 CEST 2000] Service-52 ended
[Jun 17 12:39:31 CEST 2000] Captured response -----
HTTP/1.0 302 Redirected by transform
location: http://xgate.jaxo.com/gjm/airport.rd
content-type: text/vnd.wap.wml; charset=ISO-8859-1
date: Sat, 17 Jun 2000 09:43:17 GMT
server: Netscape-Enterprise/3.5.1
```

FIGURE 26

```
[Jun 17 12:39:32 CEST 2000] Service-53 started
[Jun 17 12:39:32 CEST 2000] Service-53, Proxy: plain copy
[Jun 17 12:39:32 CEST 2000] Service-53-in: GET
http://www.aeroport.fr/scripts/WebObjects/Uccega/NTE.woa/67980000005783200000210010000083
6431/Apt_FlightsResponse.wo/3641300000736431/1/1/Mermoz HTTP/1.0
accept: text/html
proxy-connection: Keep-Alive
accept-charset: ISO-8859-1, UTF-8
host: www.aeroport.fr
user-agent: JaxoXform (1.25)

[Jun 17 12:39:33 CEST 2000] Service-53-out: HTTP/1.0 200 NeXT
content-type: text/html
date: Sat, 17 Jun 2000 09:43:23 GMT
content-length: 471
server: Netscape-Enterprise/3.5.1

[Jun 17 12:39:33 CEST 2000] Service-53-out, Xform: parsing...
[Jun 17 12:39:34 CEST 2000] Service-53-out, Xform: transforming...
[Jun 17 12:39:34 CEST 2000] Service-53-out, Xform: done.
[Jun 17 12:39:34 CEST 2000] Service-53 ended
[Jun 17 12:39:34 CEST 2000] Captured response -----
HTTP/1.0 302 Redirected by transform
location: http://xgate.jaxo.com/gjm/airport.rd
content-type: text/vnd.wap.wml; charset=ISO-8859-1
date: Sat, 17 Jun 2000 09:43:23 GMT
server: Netscape-Enterprise/3.5.1
```

FIGURE 27

```

[Jun 17 12:39:34 CEST 2000] Service-54 started
[Jun 17 12:39:34 CEST 2000] Service-54, Proxy: plain copy
[Jun 17 12:39:34 CEST 2000] Service-54-in: GET
http://www.aeroport.fr/scripts/WebObjects/Uccega/NTE.woa/67980000005783200000210010000083
6431/Apt_DayFlights.wo/7507100000936431/0.1/1/Mermoz HTTP/1.0
accept: text/html
proxy-connection: Keep-Alive
accept-charset: ISO-8859-1, UTF-8
host: www.aeroport.fr
user-agent: JaxoXform (1.25)

[Jun 17 12:39:41 CEST 2000] Service-54-out: HTTP/1.0 200 NeXT
content-type: text/html
date: Sat, 17 Jun 2000 09:43:31 GMT
content-length: 30543
server: Netscape-Enterprise/3.5.1

[Jun 17 12:39:41 CEST 2000] Service-54-out, Xform: parsing...
[Jun 17 12:39:43 CEST 2000] Service-54-out, Xform: transforming...
[Jun 17 12:39:47 CEST 2000] Service-54-out, Xform: done.
[Jun 17 12:39:47 CEST 2000] Service-54 ended
[Jun 17 12:39:47 CEST 2000] Captured response -----
HTTP/1.0 200 NeXT
content-type: text/vnd.wap.wml; charset=ISO-8859-1
date: Sat, 17 Jun 2000 09:43:31 GMT
cache-control: max-age=0
cache-control: must-revalidate
server: Netscape-Enterprise/3.5.1

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml><template><do type="prev" label="Back"><prev/></do></template>
<card title="Vols"><p>
10:50 Paris-Cdg<br/>&#183; AF7725<br/>&#183; Enregistrement D1-E2<br/>11:15
Lyon<br/>&#183; AF7862<br/>&#183; Décollé 11:23<br/>11:15 Nice<br/>&#183;
SN4882<br/>11:15 Nice<br/>&#183; FU172<br/>&#183; Enregistrement F1-F2<br/>11:55
Marseille<br/>&#183; AF5763<br/>&#183; Enregistrement H1-H2<br/>12:00 Londres
Gatwick<br/>&#183; AF5852<br/>&#183; Enregistrement I1-I2<br/>12:15 Bruxelles<br/>&#183;
SN3610<br/>&#183; Enregistrement G1-G2<br/>12:15 Bruxelles<br/>&#183; VM103<br/>14:00
Palma<br/>&#183; 2R478<br/>14:15 Paris-Cdg<br/>&#183; AF7727<br/>15:45 Paris-
Cdg<br/>&#183; AF7729<br/>15:55 Brest<br/>&#183; AF5853<br/>16:05 Malaga<br/>&#183;
JK4544<br/>16:15 Athenes<br/>&#183; RN1602<br/>17:00 Athenes<br/>&#183; EY932<br/>17:40
Marseille<br/>&#183; FU765<br/>18:35 Marseille<br/>&#183; AF5765<br/>18:40 Londres
Gatwick<br/>&#183; AF5854<br/>18:50 Londres Gatwick<br/>&#183; BA6807<br/>19:20
Nice<br/>&#183; FU176<br/></p></card></wml>

```

FIGURE 28

```

<!--
$Id: xfdoc_0.1.dtd,v 1.12 1999/05/17 15:56:26 pgr Exp $
(C) Copyright JAXO-Europe SA 1999-2000
This work contains confidential trade secrets of JAXO-Europe SA.
Use, examination, copying, transfer and disclosure to others
are prohibited, except with the express written agreement of JAXO.

xf:doc documents conversion scripts

Typical usage:

<?xml version='1.0'?>
<!DOCTYPE xf:doc PUBLIC "-//JAXO//DTD XFDOC 0.1 Draft//EN"
    "http://www.jaxo.com/DTD/xfdoc_0.1.dtd">

<xf:doc xmlns:xsl="http://www.jaxo.com/DTD/xfdoc_0.1.dtd" version='1.0'>

  <xf:template match="X">
    <!-- ECMAScript for X -->
  </xf:template match="X">

  <xf:template match="Y">
    <!-- ECMAScript for Y -->
  </xf:template match="Y">

  <xf:scripts>
    <!-- local scripts and variables, called from any template -->
  </xf:scripts>

</xf:doc>

Author:
Pierre G. Richard <pgr@jaxo.com>
-->
<!ENTITY % XF.Version "-//JAXO//DTD XFDOC 0.1 Draft//EN">

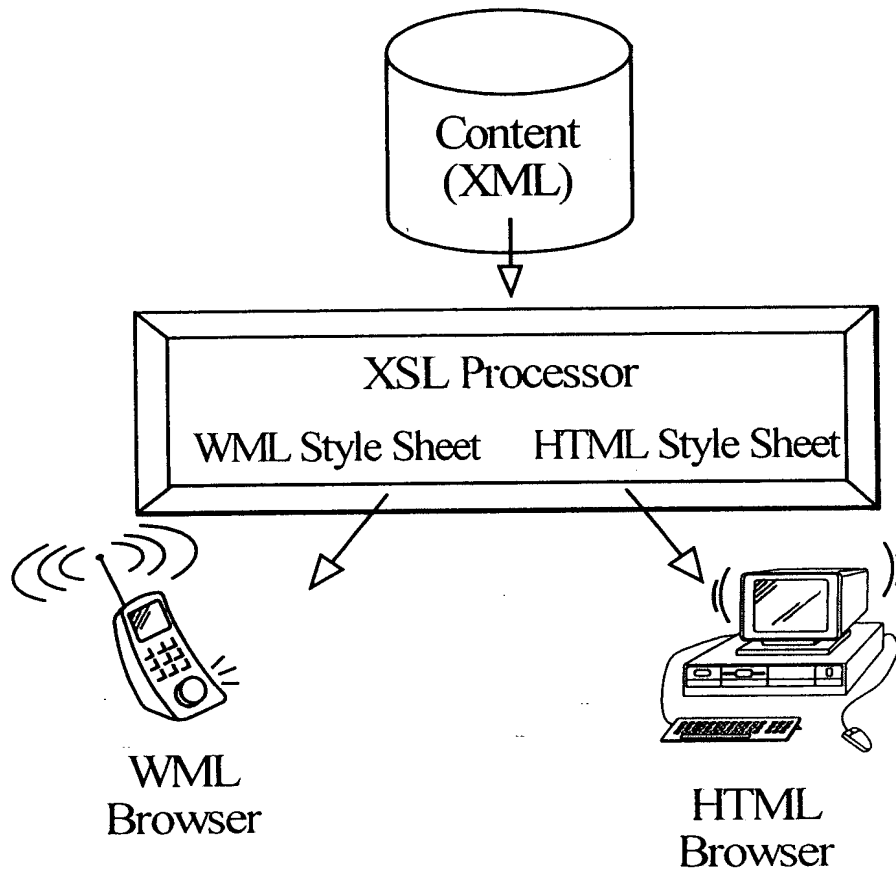
<!ELEMENT xf:doc (xf:init?, (xf:template | xf:scripts))*>
<!ATTLIST xf:doc
  in-content-type  CDATA #IMPLIED
  out-content-type CDATA #REQUIRED
  xmlns:xsl        CDATA #IMPLIED
  version          CDATA #IMPLIED
>

<!ELEMENT xf:template CDATA>
<!ATTLIST xf:template
  match  CDATA #IMPLIED
  name   CDATA #IMPLIED
>

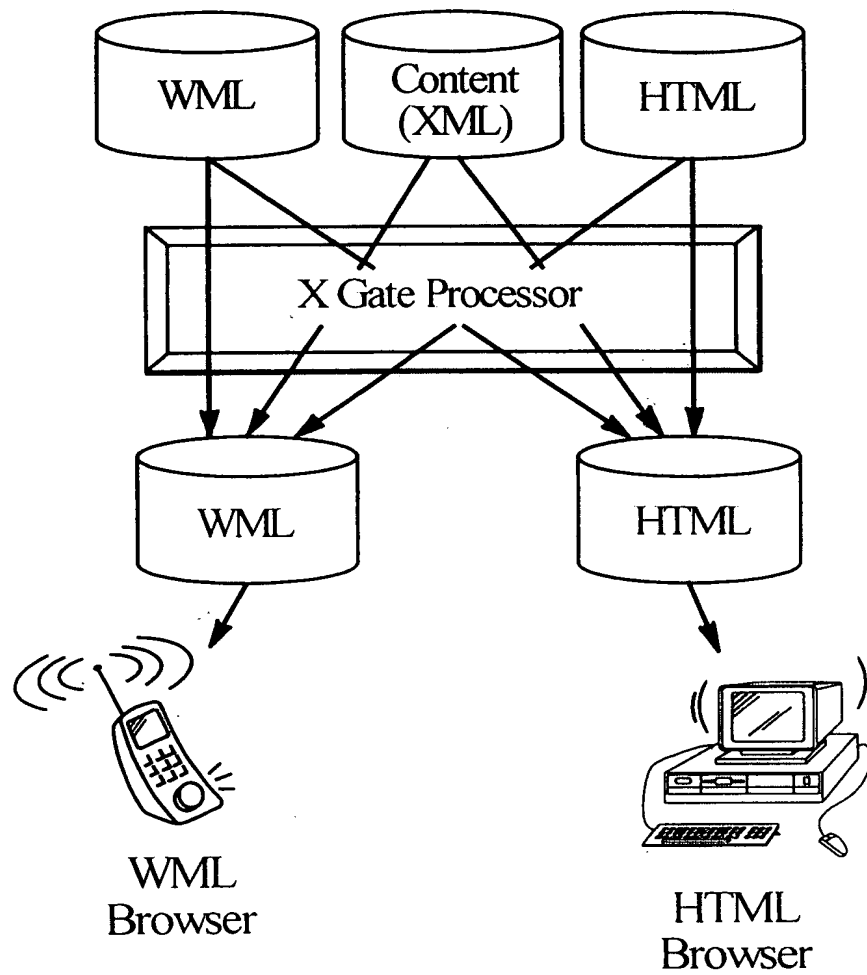
<!ELEMENT xf:scripts CDATA>
<!ELEMENT xf:init     CDATA>

```

FIG.1



2 / 28

FIG. 2

3 / 28

FIG. 3

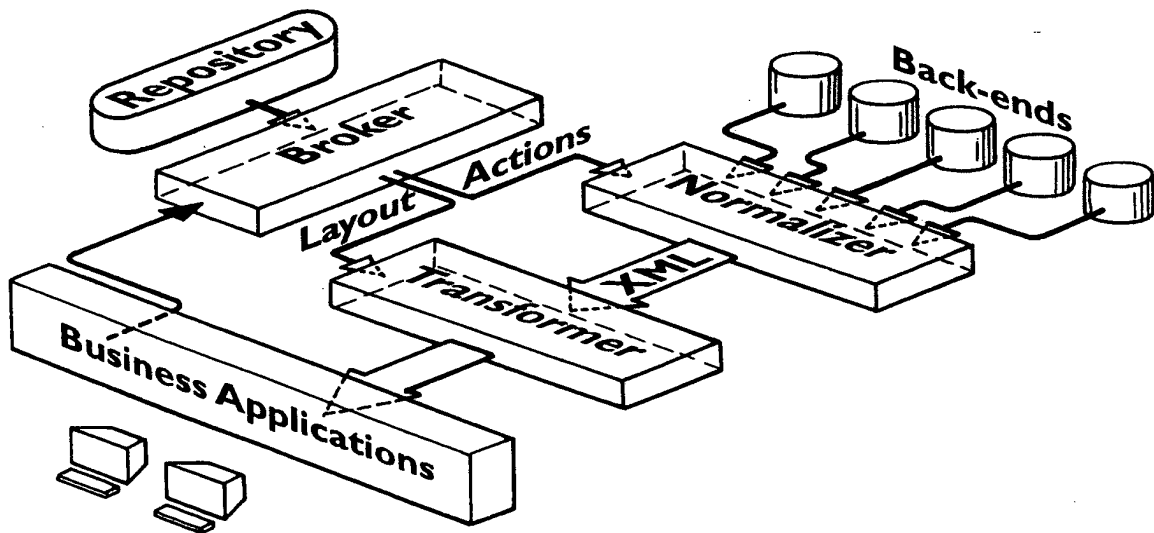


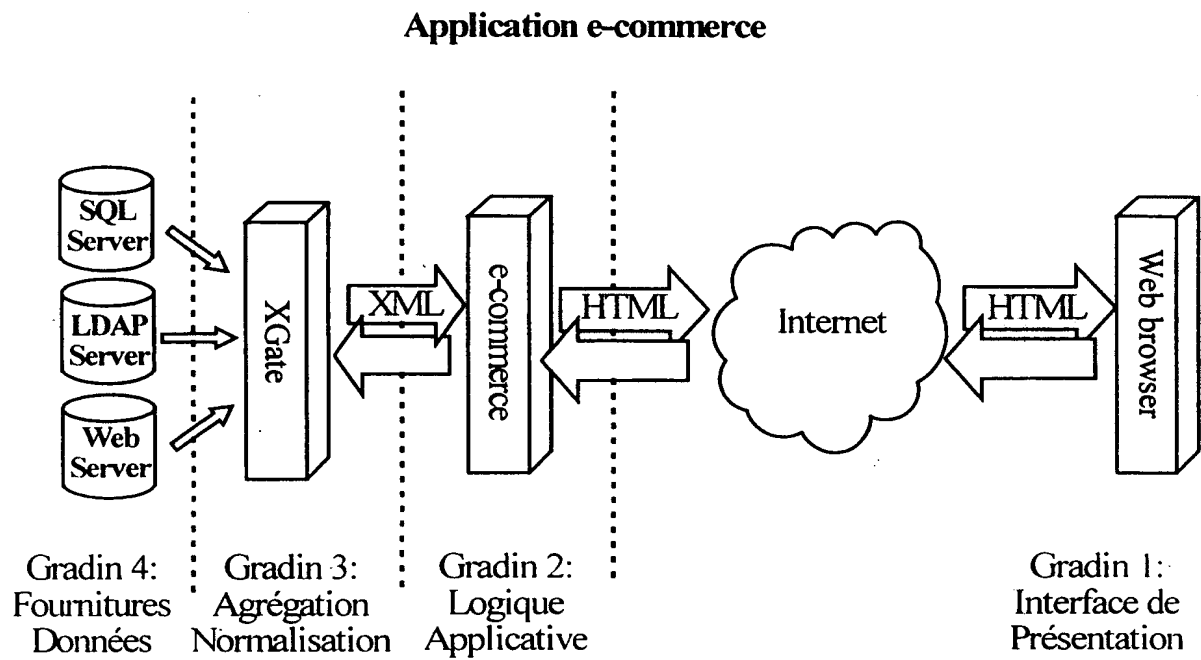
FIG. 4

FIG.5

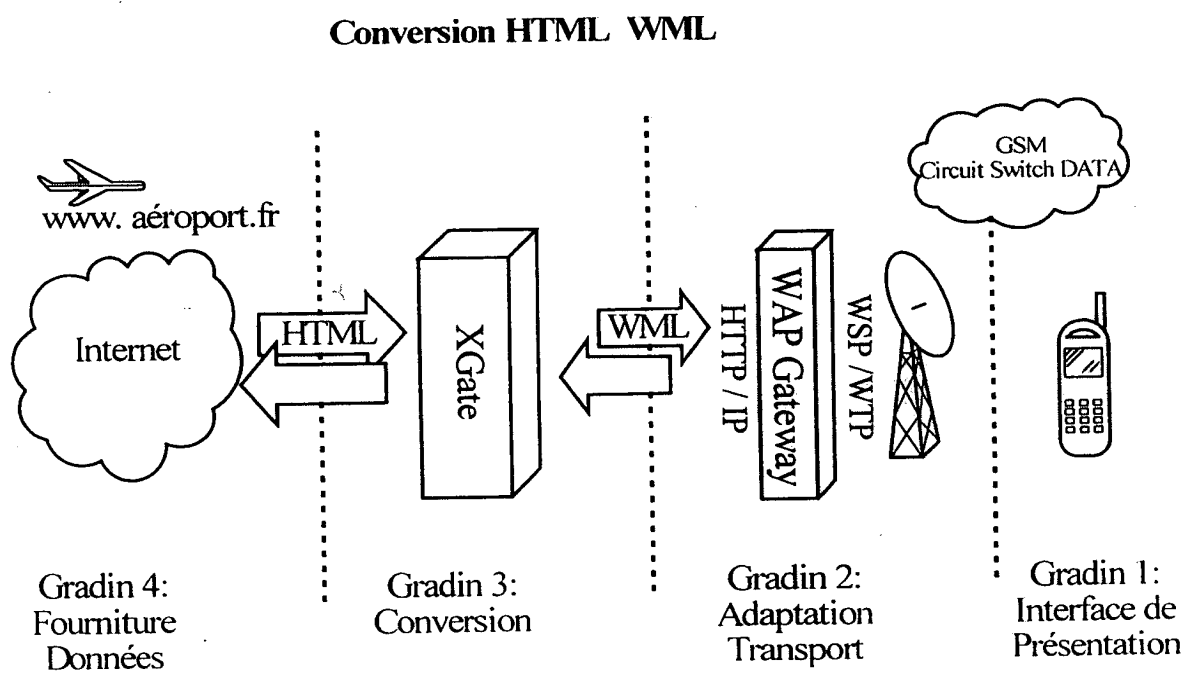
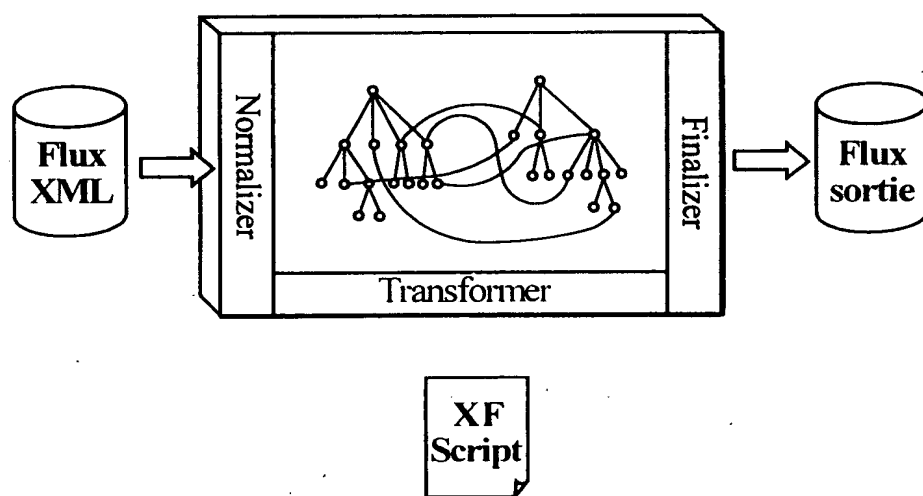


FIG. 6

Transformation : flux



7/28

FIG. 7

AGREGATION ET TRANSFORMATION

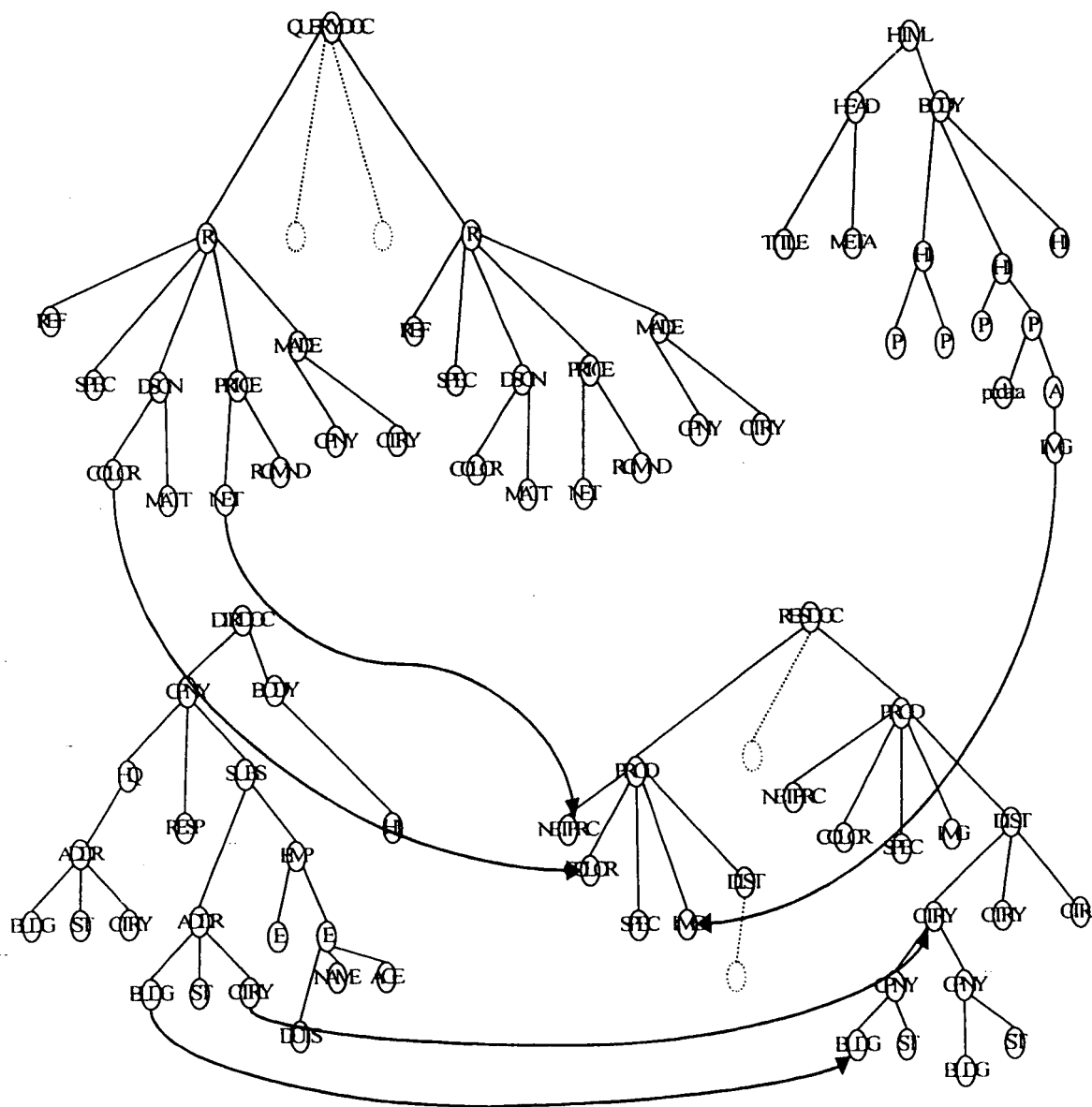


FIG-8

<xf:template match=□P□>
(procédure P)
</xf:template>

9 / 28

FIG. 9

```
<xf:template match="HTML">  
  XF.trace();  
  XF.applyTemplates("origin().descendant(all,(FRAMESET | BODY))");  
</xf:template>
```

10 / 28

FIG_10

```
<xf:template match="FRAMESET">
  XF.setVar("framed", 1);
  var list = this.getChildNodes();
  for (var i=0; ;++i) {
    var child = list.item(i);
    if (child == null) break;
    if (child.getNodeName() == "FRAME") {
      var name = child.getAttribute("name");
      if (
        ((name == null) && (child.getNextSibling() == null)) ||
        (name == "HOME") || (name == "MainMenu") ||
        (name == "Content") || (name == "Result")
      ) {
        XF.redirectTo(child.getAttribute("src"));
        break;
      }
    }
  }
}</xf:template>
```

101

102

11 / 28

FIG.11

```
<xf:template name="body" match="BODY"
  if (null == XF.getVar("framed")) {
    XF.result().write(prolog);
    var url = XF.getURL();
    if (url.indexOf("DayFlight") > 0) {
      XF.applyTemplates("origin().descendant(all,TBODY)");
    } else {
      XF.applyTemplates("origin().descendant(all,IMG)");
    }
    XF.result().write(epilog);
  }
</xf:template>
```

113

111

// result 114

// HomePage (Dep/Arr)

115

112

12 / 28

FIG.12

<pre><?xml version="1.0"?> <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml"> <wml> <card> <p> Départs Arrivées </p> </card> </wml></pre>	<p>prologue</p> <p>epilogue</p>
---	---------------------------------

```
<xf:template name="menuItem" match="IMG43">
  var text = this.getAttribute("src");
  if (text.indexOf("DEPART") > 0) {
    addAnchor(this, "Départs", "airport.21");
  }else if (text.indexOf("ARRIVE") > 0) {
    addAnchor(this, "Arrivées", "airport.22");
  }
</xf:template>
```

14 / 28

FIG_14

```
<xf:scripts>
function addAnchor(node, text, localRef) {
  while (null != (node = node.getParentNode())) {
    if (node.getNodeName() == 'A') {
      XF.result().writeAnchor(text, node.getAttribute("href"), localRef)
      break;
    }
  }
}
</xf:scripts>
```

15 / 28

FIG.15

Texte	Départs
Réf hypertexte	http://www.montpellier.aeroport.fr/cgi-bin/WebObjects.dll/Uccega/MPL?action=goDayDepartures
Réf locale	airport.21

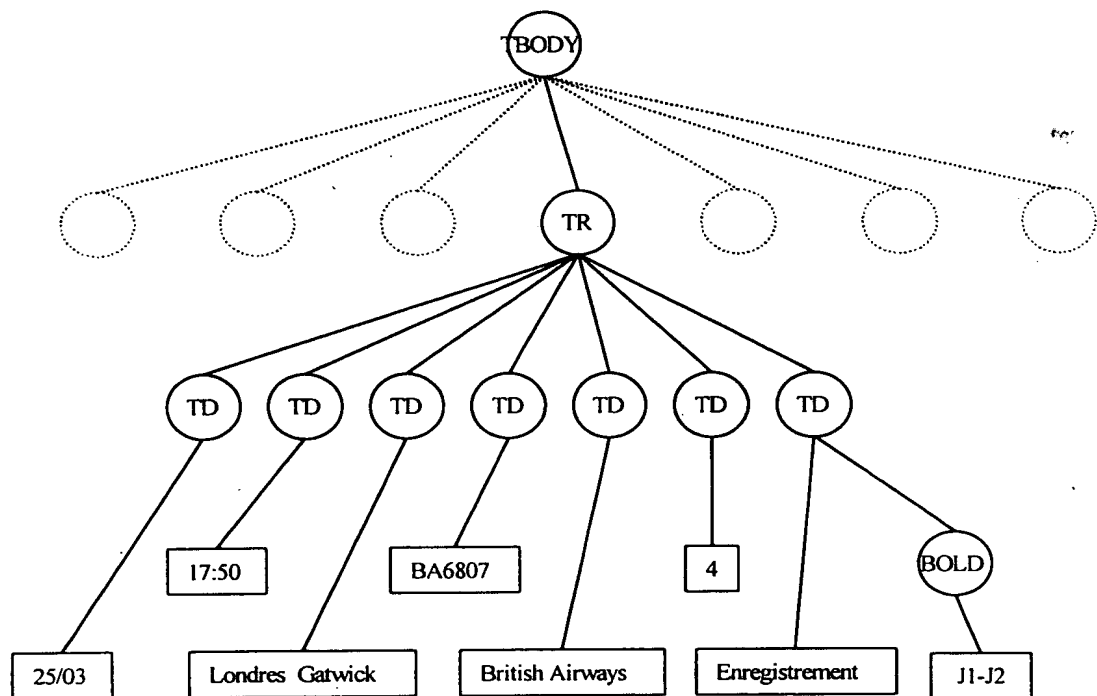
FIG.16**Affichage du tableau des horaires**

FIG. 17

```
<xf:template match="TBODY">
  var list = this.getChildNodes();
  for (var i=0; ;++i) {
    child = list.item(i);
    if (child == null) break;
    if (child.getAttribute("bgcolor") == "FFFFFF") { // irrelevant
      i += 3;
    } else {
      var data = new FlightData();
      XF.applyTemplates(
        "origin().child(" + i + ").descendant(all,#text)",
        data
      );
      XF.result().write(data.timeFrom + "<br/>" + data.others);
    }
  }
</xf:template>
```

171

173

172

174

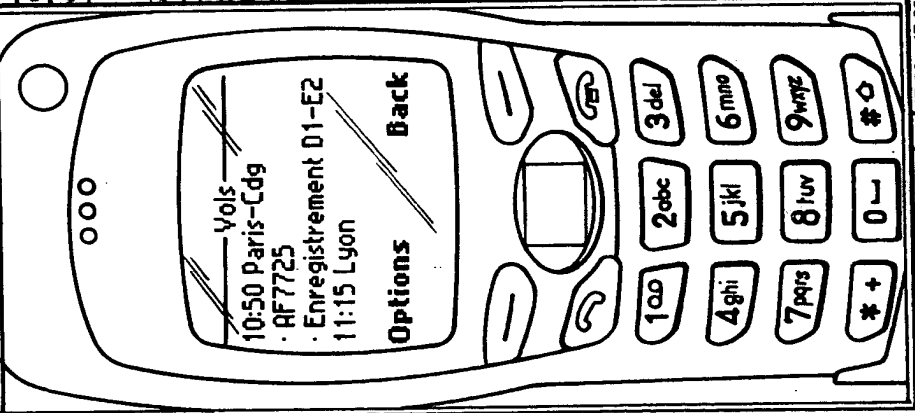
FIG_18

La procédure template pour chaque portion de texte s'écrit simplement :

```
<xf:template name="flight" match="text()">
  var text = this.getNodeValue().reduce();
  if (text.length > 0) {
    var i = arg0.item++;
    if (i == 1) { arg0.timeFrom = text + " "; // arrival time
    else if (i == 2) arg0.timeFrom += text; // from
    else if (i == 3) arg0.others = "." + text + "<br/>"; // airline
    else if (i == 4) arg0.others += "." + text + "<br/>"; // buildg
    else if (i == 5) arg0.others += "." + text + "<br/>"; // remarks
    else if (i == 6) arg0.others += "." + text + "<br/>"; // departure time
  }
</xf:template>
```

FIG_19

19 / 28



report:fr/scripts/WebObjects/Uccage/NTE?action=goDayDepartures-Microsoft Internet Explorer

www.aeroport.fr/scripts/WebObjects/Uccage/NTE?action=goDayDepartures

Departs du jour

Samedi 17 Juin 11:39

REPORTS FRANCAIS

Atique : Les départs

fr:franche horaire - provenance - compagnie | Autre recherche

DESTINATION	N° VOL	COMPAGNIE	SALLE DEPART	OBSERVATIONS	HEURE ARRIVEE
Paris-Cdg	AF7725	Air France	2	Enregistrement D1-E2	
Lyon	AF7862	Air France	3B	Décollé 11:23	
Nice	SN4882	Sabena	1		
Nice	EU172	Air Littoral	1	Enregistrement F1-F2	
Marseille	AF5763	Air France	1	Enregistrement H1-H2	
Londres Gatwick	AF5852	Air France	4	Enregistrement H1-H2	
Bruxelles	SN3610	Sabena	1	Enregistrement G1-G2	
Bruxelles	VM103	Regional Airlines	1		
Palma	2R478	Star Airlines	5		
Paris-Cdg	AF7727	Air France	3A		

DESTINATION	N° VOL	COMPAGNIE	SALLE DEPART	OBSERVATIONS	HEURE ARRIVEE
Paris-Cdg	AF7729	Air France	2		
Brest	AF5853	Air France	2		
Malaga	JK4544	Spanair, Sa	5		
Athènes	RN1602	Euralair International	3A		
Athènes	EY932	Aerolyon	5		
Marseille	EU765	Air Littoral	1		
Marseille	AF5765	Air France	1		
Londres Gatwick	AF5854	Air France	4		
Londres Gatwick	BA6807	British Airways	4		
Nice	EU176	Air Littoral	1		

Ces horaires ont été mis à jour le 17/06/2000 à 11h30. Pour Internet

Les vols suivants

17.06 17:00
17.06 17:30
17.06 18:35
17.06 18:40
17.06 18:50
17.06 19:20

20 / 28

FIG. 20

[Jun 17 12:38:55 CEST 2000] Service-47 started
[Jun 17 12:38:55 CEST 2000] Service-47, Proxy: plain copy
[Jun 17 12:38:55 CEST 2000] Service-47-in: GET http://www.Nantes.aeroport.fr HTTP/1.0
accept: text/html
proxy-connection: Keep-Alive
accept-charset: ISO-8859-1, UTF-8
host: www.Nantes.aeroport.fr
user-agent: JaxoXform (1.25)

[Jun 17 12:38:56 CEST 2000] Service-47-out: HTTP/1.0 200 OK
content-type: text/html

date: Sat, 17 Jun 2000 09:42:47 GMT

last-modified: Fri, 12 May 2000 14:53:47 GMT

accept-ranges: bytes

link: <http://www.aeroport.fr/?PageServices>; rel="PageServices"

content-length: 5932

server: Netscape-Enterprise/3.5.1

[Jun 17 12:38:56 CEST 2000] Service-47-out, Xform: parsing...

Cannot start element "META": a required element is missing in BODY
line 3, col 76

in: "Content-Type" CONTENT="text/html; charset=iso-8859-1">

at: >-----*

Cannot start element "META": a required element is missing in BODY
line 4, col 451

in: 're, terminal, piste, runway, fret aerien, air freight">

at: >-----*

Cannot start element "META": a required element is missing in BODY
line 5, col 403

in: 'mercial services, tourism in the Aquitaine region ..."> </head>

at: >-----*

Warning: invalid end tag: "HEAD"

line 5, col 413

in: 'rvices, tourism in the Aquitaine region ..."> </head>

at: >-----*

Warning: invalid end tag: "NOFRAMES"

line 60, col 12

in: '</noframes>

at: >-----*

[Jun 17 12:38:57 CEST 2000] Service-47-out, Xform: transforming...

[Jun 17 12:38:57 CEST 2000] Service-47-out, Xform: done.

[Jun 17 12:38:57 CEST 2000] Service-47 ended

[Jun 17 12:38:57 CEST 2000] Captured response -----

HTTP/1.0 302 Redirected by transform

location: http://xgate.jaxo.com/gjm/airport.rd

content-type: text/vnd.wap.wml; charset=ISO-8859-1

date: Sat, 17 Jun 2000 09:42:47 GMT

last-modified: Fri, 12 May 2000 14:53:47 GMT

accept-ranges: bytes

link: <http://www.aeroport.fr/?PageServices>; rel="PageServices"

server: Netscape-Enterprise/3.5.1

21 / 28

FIG. 21

```
[Jun 17 12:38:58 CEST 2000] Service-48 started
[Jun 17 12:38:58 CEST 2000] Service-48, Proxy: plain copy
[Jun 17 12:38:58 CEST 2000] Service-48-in: GET http://www.nantes.aeroport.fr/scripts/WebObjects/Uccega/NTE
HTTP/1.0
accept: text/html
proxy-connection: Keep-Alive
accept-charset: ISO-8859-1, UTF-8
host: www.nantes.aeroport.fr
user-agent: JaxoXform (1.25)

[Jun 17 12:38:59 CEST 2000] Service-48-out: HTTP/1.0 200 NeXT
content-type: text/html
date: Sat, 17 Jun 2000 09:42:49 GMT
content-length: 321
server: Netscape-Enterprise/3.5.1

[Jun 17 12:38:59 CEST 2000] Service-48-out, Xform: parsing...
[Jun 17 12:38:59 CEST 2000] Service-48-out, Xform: transforming...
[Jun 17 12:38:59 CEST 2000] Service-48-out, Xform: done.
[Jun 17 12:38:59 CEST 2000] Service-48 ended
[Jun 17 12:38:59 CEST 2000] Captured response -----
HTTP/1.0 302 Redirected by transform
location: http://xgate.jaxo.com/gjm/airport.rd
content-type: text/vnd.wap.wml; charset=ISO-8859-1
date: Sat, 17 Jun 2000 09:42:49 GMT
server: Netscape-Enterprise/3.5.1
```

FIG. 22

```
[Jun 17 12:38:59 CEST 2000] Service-49 started
[Jun 17 12:39:00 CEST 2000] Service-49, Proxy: plain copy
[Jun 17 12:39:00 CEST 2000] Service-49-in: GET
http://www.nantes.aeroport.fr/scripts/WebObjects/Uccega/NTE.woa/20932000002680200000233110
0000536431/Main.wo/7192100000436431/1/1/Mermoz HTTP/1.0
accept: text/html
proxy-connection: Keep-Alive
accept-charset: ISO-8859-1, UTF-8
host: www.nantes.aeroport.fr
user-agent: JaxoXform (1.25)

[Jun 17 12:39:05 CEST 2000] Service-49-out: HTTP/1.0 200 NeXT
content-type: text/html
date: Sat, 17 Jun 2000 09:42:52 GMT
content-length: 490
server: Netscape-Enterprise/3.5.1

[Jun 17 12:39:05 CEST 2000] Service-49-out, Xform: parsing...
[Jun 17 12:39:05 CEST 2000] Service-49-out, Xform: transforming...
[Jun 17 12:39:05 CEST 2000] Service-49-out, Xform: done.
[Jun 17 12:39:05 CEST 2000] Service-49 ended
[Jun 17 12:39:05 CEST 2000] Captured response -----
HTTP/1.0 302 Redirected by transform
location: http://xgate.jaxo.com/gjm/airport.rd
content-type: text/vnd.wap.wml; charset=ISO-8859-1
date: Sat, 17 Jun 2000 09:42:52 GMT
server: Netscape-Enterprise/3.5.1
```

23 / 28

FIG. 23

[Jun 17 12:39:05 CEST 2000] Service-50 started
[Jun 17 12:39:06 CEST 2000] Service-50, Proxy: plain copy
[Jun 17 12:39:06 CEST 2000] Service-50-in: GET http://www.aeroport.fr/Uccega/NTE/Html/FR/HP/home.htm
HTTP/1.0
accept: text/html
proxy-connection: Keep-Alive
accept-charset: ISO-8859-1, UTF-8
host: www.aeroport.fr
user-agent: JaxoXform (1.25)

[Jun 17 12:39:10 CEST 2000] Service-50-out: HTTP/1.0 200 OK
content-type: text/html
date: Sat, 17 Jun 2000 09:42:57 GMT
last-modified: Wed, 17 May 2000 16:10:48 GMT
accept-ranges: bytes
link: <http://www.aeroport.fr/Uccega/NTE/Html/FR/HP/home.htm?PageServices>; rel="PageServices"
content-length: 507
server: Netscape-Enterprise/3.5.1

[Jun 17 12:39:10 CEST 2000] Service-50-out, Xform: parsing...
Cannot start element "BODY": not permitted by the document structure.
line 11, col 35
in: '<noframes><body bgcolor="#FFFFFF">'
at: >-----*

[Jun 17 12:39:11 CEST 2000] Service-50-out, Xform: transforming...
[Jun 17 12:39:11 CEST 2000] Service-50-out, Xform: done.
[Jun 17 12:39:11 CEST 2000] Service-50 ended
[Jun 17 12:39:11 CEST 2000] Captured response -----
HTTP/1.0 302 Redirected by transform
location: http://xgate.jaxo.com/gjm/airport.rd
content-type: text/vnd.wap.wml; charset=ISO-8859-1
date: Sat, 17 Jun 2000 09:42:57 GMT
last-modified: Wed, 17 May 2000 16:10:48 GMT
accept-ranges: bytes
link: <http://www.aeroport.fr/Uccega/NTE/Html/FR/HP/home.htm?PageServices>; rel="PageServices"
server: Netscape-Enterprise/3.5.1

24/28

FIG. 24

```
[Jun 17 12:39:11 CEST 2000] Service-51 started
[Jun 17 12:39:12 CEST 2000] Service-51, Proxy: plain copy
[Jun 17 12:39:12 CEST 2000] Service-51-in: GET http://www.aeroport.fr/Uccega/NTE/Html/FR/HP/2.htm
HTTP/1.0
accept: text/html
proxy-connection: Keep-Alive
accept-charset: ISO-8859-1, UTF-8
host: www.aeroport.fr
user-agent: JaxoXform (1.25)

[Jun 17 12:39:12 CEST 2000] Service-51-out: HTTP/1.0 200 OK
content-type: text/html
date: Sat, 17 Jun 2000 09:43:03 GMT
last-modified: Wed, 17 May 2000 16:10:48 GMT
accept-ranges: bytes
link: <http://www.aeroport.fr/Uccega/NTE/Html/FR/HP/2.htm?PageServices>; rel="PageServices"
content-length: 2603
server: Netscape-Enterprise/3.5.1

[Jun 17 12:39:13 CEST 2000] Service-51-out, Xform: parsing...
Cannot start element "BODY": a required element is missing in TITLE
  line 10, col 23
  in: '<BODY bgColor=#ffffff>'
  at: >-----*

[Jun 17 12:39:13 CEST 2000] Service-51-out, Xform: transforming...
[Jun 17 12:39:13 CEST 2000] Service-51-out, Xform: done.
[Jun 17 12:39:13 CEST 2000] Service-51 ended
[Jun 17 12:39:13 CEST 2000] Captured response -----
HTTP/1.0 200 OK
content-type: text/vnd.wap.wml; charset=ISO-8859-1
date: Sat, 17 Jun 2000 09:43:03 GMT
last-modified: Wed, 17 May 2000 16:10:48 GMT
accept-ranges: bytes
link: <http://www.aeroport.fr/Uccega/NTE/Html/FR/HP/2.htm?PageServices>; rel="PageServices"
server: Netscape-Enterprise/3.5.1
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml><template><do type="prev" label="Back"><prev/></do></template>
<card title="Montpellier"><p>
<a href="/$(sid)/airport.21">Départs</a>
<a href="/$(sid)/airport.22">Arrivées</a></p></card></wml>
```

25 / 28

FIG. 25

```
[Jun 17 12:39:18 CEST 2000] Service-52 started
[Jun 17 12:39:18 CEST 2000] Service-52, Proxy: plain copy
[Jun 17 12:39:18 CEST 2000] Service-52-in: GET
http://www.aeroport.fr/scripts/WebObjects/Uccega/NTE?action=goDayDepartures HTTP/1.0
accept: text/html
proxy-connection: Keep-Alive
accept-charset: ISO-8859-1, UTF-8
host: www.aeroport.fr
user-agent: JaxoXform (1.25)

[Jun 17 12:39:26 CEST 2000] Service-52-out: HTTP/1.0 200 NeXT
content-type: text/html
date: Sat, 17 Jun 2000 09:43:17 GMT
content-length: 563
server: Netscape-Enterprise/3.5.1

[Jun 17 12:39:27 CEST 2000] Service-52-out, Xform: parsing...
[Jun 17 12:39:31 CEST 2000] Service-52-out, Xform: transforming...
[Jun 17 12:39:31 CEST 2000] Service-52-out, Xform: done.
[Jun 17 12:39:31 CEST 2000] Service-52 ended
[Jun 17 12:39:31 CEST 2000] Captured response -----
HTTP/1.0 302 Redirected by transform
location: http://xgate.jaxo.com/gjm/airport.rd
content-type: text/vnd.wap.wml; charset=ISO-8859-1
date: Sat, 17 Jun 2000 09:43:17 GMT
server: Netscape-Enterprise/3.5.1
```

26/28

FIG. 26

```
[Jun 17 12:39:32 CEST 2000] Service-53 started
[Jun 17 12:39:32 CEST 2000] Service-53, Proxy: plain copy
[Jun 17 12:39:32 CEST 2000] Service-53-in: GET
http://www.aeroport.fr/scripts/WebObjects/Uccega/NTE.woa/679800000057832000002100100000836431/
Apt_FlightsResponse.wo/3641300000736431/1/1/Mermoz HTTP/1.0
accept: text/html
proxy-connection: Keep-Alive
accept-charset: ISO-8859-1, UTF-8
host: www.aeroport.fr
user-agent: JaxoXform (1.25)

[Jun 17 12:39:33 CEST 2000] Service-53-out: HTTP/1.0 200 NeXT
content-type: text/html
date: Sat, 17 Jun 2000 09:43:23 GMT
content-length: 471
server: Netscape-Enterprise/3.5.1

[Jun 17 12:39:33 CEST 2000] Service-53-out, Xform: parsing...
[Jun 17 12:39:34 CEST 2000] Service-53-out, Xform: transforming...
[Jun 17 12:39:34 CEST 2000] Service-53-out, Xform: done.
[Jun 17 12:39:34 CEST 2000] Service-53 ended
[Jun 17 12:39:34 CEST 2000] Captured response -----
HTTP/1.0 302 Redirected by transform
location: http://xgate.jaxo.com/gjm/airport.rd
content-type: text/vnd.wap.wml; charset=ISO-8859-1
date: Sat, 17 Jun 2000 09:43:23 GMT
server: Netscape-Enterprise/3.5.1
```

27/28

FIG. 27

```
[Jun 17 12:39:34 CEST 2000] Service-54 started
[Jun 17 12:39:34 CEST 2000] Service-54, Proxy: plain copy
[Jun 17 12:39:34 CEST 2000] Service-54-in: GET
http://www.aeroport.fr/scripts/WebObjects/Uccega/NTE.woa/679800000057832000002100100000836431/Apt_DayFlag-
hts.wa/7507100000936431/0.1/1/Mermoz HTTP/1.0
accept: text/html
proxy-connection: Keep-Alive
accept-charset: ISO-8859-1, UTF-8
host: www.aeroport.fr
user-agent: laxoXform (1.25)

[Jun 17 12:39:41 CEST 2000] Service-54-out: HTTP/1.0 200 NeXT
content-type: text/html
date: Sat, 17 Jun 2000 09:43:31 GMT
content-length: 30543
server: Netscape-Enterprise/3.5.1

[Jun 17 12:39:41 CEST 2000] Service-54-out, Xform: parsing...
[Jun 17 12:39:43 CEST 2000] Service-54-out, Xform: transforming...
[Jun 17 12:39:47 CEST 2000] Service-54-out, Xform: done.
[Jun 17 12:39:47 CEST 2000] Service-54 ended
[Jun 17 12:39:47 CEST 2000] Captured response -----
HTTP/1.0 200 NeXT
content-type: text/vnd.wap.wml; charset=ISO-8859-1
date: Sat, 17 Jun 2000 09:43:31 GMT
cache-control: max-age=0
cache-control: must-revalidate
server: Netscape-Enterprise/3.5.1

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml><template><do type="prev" label="Back"><prev/></do></template>
<card title="Vols"><p>
10:50 Paris-Cdg<br/>&#183; AF7725<br/>&#183; Enregistrement D1-E2<br/>11:15 Lyon<br/>&#183;
AF7862<br/>&#183; Décollé 11:23<br/>11:15 Nice<br/>&#183; SN4882<br/>11:15 Nice<br/>&#183;
FU172<br/>&#183; Enregistrement F1-F2<br/>11:55 Marseille<br/>&#183; AF5763<br/>&#183; Enregistrement
H1-H2<br/>12:00 Londres Gatwick<br/>&#183; AF5852<br/>&#183; Enregistrement I1-I2<br/>12:15
Bruxelles<br/>&#183; SN3610<br/>&#183; Enregistrement G1-G2<br/>12:15 Bruxelles<br/>&#183;
VM103<br/>14:00 Palma<br/>&#183; 2R478<br/>14:15 Paris-Cdg<br/>&#183; AF7727<br/>15:45 Paris-
Cdg<br/>&#183; AF7729<br/>15:55 Brest<br/>&#183; AF5853<br/>16:05 Malaga<br/>&#183;
JK4544<br/>16:15 Athenes<br/>&#183; RN1602<br/>17:00 Athenes<br/>&#183; EY932<br/>17:40
Marseille<br/>&#183; FU765<br/>18:35 Marseille<br/>&#183; AF5765<br/>18:40 Londres Gatwick<br/>&#183;
AF5854<br/>18:50 Londres Gatwick<br/>&#183; BA6807<br/>19:20 Nice<br/>&#183;
FU176<br/></p></card></wml>
```


FIG_28

```

<!--
$Id: xfdoc_0.1.dtd,v 1.12 1999/05/17 15:56:26 pgr Exp $
(C) Copyright JAXO-Europe SA 1999-2000
This work contains confidential trade secrets of JAXO-Europe SA.
Use, examination, copying, transfer and disclosure to others
are prohibited, except with the express written agreement of JAXO.

xf:doc documents conversion scripts

Typical usage:

<?xml version='1.0'?>
<!DOCTYPE xf:doc PUBLIC "-//JAXO//DTD XFDOC 0.1 Draft//EN"
    "http://www.jaxo.com/DTD/xfdoc_0.1.dtd">

<xf:doc xmlns:xsl="http://www.jaxo.com/DTD/xfdoc_0.1.dtd" version='1.0'>

  <xf:template match="X">
    <!-- ECMAScript for X -->
  </xf:template match="X">

  <xf:template match="Y">
    <!-- ECMAScript for Y -->
  </xf:template match="Y">

  <xf:scripts>
    <!-- local scripts and variables, called from any template -->
  </xf:scripts>

</xf:doc>

Author:
Pierre G. Richard <pgr@jaxo.com>
-->
<ENTITY % XF.Version "-//JAXO//DTD XFDOC 0.1 Draft//EN">

<ELEMENT xf:doc (xf:init?, (xf:template | xf:scripts))*>
<!ATTLIST xf:doc
  in-content-type CDATA #IMPLIED
  out-content-type CDATA #REQUIRED
  xmlns:xsl CDATA #IMPLIED
  version CDATA #IMPLIED
>

<ELEMENT xf:template CDATA>
<!ATTLIST xf:template
  match CDATA #IMPLIED
  name CDATA #IMPLIED
>

<ELEMENT xf:scripts CDATA>
<ELEMENT xf:init CDATA>

```

THIS PAGE BLANK (USPTO)